Engineering a Better Receptor: Characterization of Retinoid X Receptor Alpha and Functional Variants

A Dissertation
Presented To
The Academic Faculty

By

Terry J. Watt

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in the
School of Chemistry and Biochemistry

Georgia Institute of Technology

December 2007

Engineering a Better Receptor: Characterization of Retinoid X Receptor Alpha and Functional Variants

Approved by:

Dr. Donald F. Doyle, advisor
School of Chemistry & Biochemistry
Georgia Institute of Technology

Dr. Nicholas V. Hud
School of Chemistry & Biochemistry
Georgia Institute of Technology

Dr. Julia Kubanek
School of Chemistry & Biochemistry
School of Biology
Georgia Institute of Technology

Dr. Stephen C. Harvey
School of Biology
School of Chemistry & Biochemistry, adjunct
Georgia Institute of Technology

Dr. Andreas S. Bommarius
School of Chemical & Biomolecular Engineering
School of Chemistry & Biochemistry, adjunct
Georgia Institute of Technology

Date Approved: 31 October, 2007

To Shannon

ACKNOWLEDGMENTS

This dissertation is the result of many people's efforts. Although it is my name on this document, the credit for it deserves to be shared much more widely, and I've done my best to recognize everyone who contributed. In the event I missed someone, you have my sincerest apologies, and may whack me at your earliest convenience.

Without the use of equipment from many labs, very little of this work would have been accomplished. I thank the Williams lab for use of their fluorimeter, calorimeter, electrophoresis rigs, and microscopes; the Hud lab for use of their calorimeter, circular dichroism spectropolarimeter, dynamic light scattering instrument, and electrophoresis rig; the Fahrni lab for use of their HPLC; the Eckert-Liotta and Teja labs for use of their densimeter; and the Bommarius lab for providing many small pieces of equipment and reagents. Michael Beck, Dr. James Broering, Dr. John Cody, Aaron Engelhart, Eric Horowitz, Dr. Swapan Jain, Ozgul Persil, Dr. Tracey Thaler, Dr. Igor Vilfan, and Derrick Watkins were all very helpful when I needed instrument training or had far too many questions.

Of equal importance are those who provided reagents and ran numerous samples. Thanks go to Dr. Matt Redinbo (University of North Carolina) for providing the plasmids containing PXR and an SRC-1 fragment; the Chernoff lab, and especially Gary Newnam, for yeast expression reagents; Dr. David Botswick and Dr. Cameron Sullards, for running hundreds of samples and helping me work through some unusual mass spectrometry experiments; Dr. Nadia Bouguslavsky for synthesizing the LxxLL peptide with truly amazing yield; Dr. Jing Li for providing the nonsense counterpart of the LxxLL peptide; Dr. Hyuk-Kyu Seoh (Georgia State University) for running many samples in the analytical ultracentrifuge,

and spending too much of his time answering my questions; Dr. Pehr Harbury (Stanford University) for the MPAX reagents; and Dr. Andrew Shulman (University of Texas, Southwestern Medical Center) for providing raw nuclear receptor alignments. In addition, for taking the time to provide thorough answers to requests for experimental details, I thank Dr. Thomas Consler (GlaxoSmithKline) and Dr. Mark Harder (Oregon State University).

Many people also contributed in ways that were less tangible, but I'd be much worse without them. A huge thank-you goes to Tatsuya "T" Maehigashi, Dr. Karen Pollizi, and Dr. Javier Chaparro-Riggers for many hours of enlightening, and enjoyable, conservation. Eduardo Vazquez-Figueroa, Karl Hüettinger, Dr. Bernard Loo, Reagan McRae, and the other members of the Bommarius and Fahrni labs also provided help and moral support. Kathy Huggins and Dr. Cam Tyson put in tremendous effort on behalf of the graduate student body. Allen Echols was always available when our lab needed repairs or upgrades. I thank Dr. Marc Borodovsky, Dr. Paul Richardson, and Dr. Steve Harvey for their valuable feedback on the ESPSearch manuscript, as well as Biotechniques for allowing me to reprint it in this dissertation. Though I made very poor use of them, my committee members managed to be helpful anyway: Dr. Andreas Bommarius, Dr. Steve Harvey, Dr. Nick Hud, and Dr. Julia Kubanek. Very little is accomplished without funding; the Seaver Foundation, Research Corp., the National Institutes of Health, the National Science Foundation (via the Biology IGERT program), the IBB Undergraduate Research Scholars program, the American Society for Engineering Education, and the Department of Defense all provided funds that allowed me to work.

Everyone in the Doyle lab, past and present, has contributed to this project. Dr. Bahareh Azizi taught me molecular biology when I arrived in the lab without any training, developed the technique of chemical

complementation, performed some cloning and yeast assays when the payoff looked to quite remote, and contributed in numerous small ways. Dr. Lauren Schwimmer developed the libraries, found most of the variants I characterized, and provided many hours of assistance. Dr. Priyanka Rohatgi performed several cell culture experiments at my request. Kenyetta Johnson shared plasmids and information on a wide variety of subjects. Amanda Ousley cloned hRXRα(CDE) into pET28 with rather more effort than she initially expected. These, and the other members of the lab (Hilda Castillo, Anna Duraj-Thatte, James Kratzer, Hally Shaffer, and Jennifer Taylor), suffered through presentation practices and paper drafts, and are starting to turn some of those "future work" ideas into reality.

My advisor, Dr. Donald Doyle, deserves much of the credit for this work. Despite the various difficulties that arose in his life and career during my tenure as a graduate student, he has always provided an environment in which I could be productive. And although there were times when neither of us felt like there was enough time to accomplish what needed to be done, he somehow managed to provide sufficient feedback. The degree of trust he has in his students is immensely valuable. For every experiment he insisted I try when I resisted, he'd allow me to spend many times the effort on experiments of my devising, even when I seemed to be spending all my time chasing shadows.

I cannot possibly express how much credit for this work truly belongs to the undergraduates I have worked with. Suhit Patel, although he only spent a few weeks in the lab, still made significant contributions in cloning the hRXRα variants lacking cysteines and in setting up hundreds of crystallography trials. Nicholas Drahush performed many hours of experiments even when he had to simply operate on faith that I had any idea what the value was in doing so. He stuck with a project that turned out

quite different from what I'd originally sold him on, and generated significant portion of the data included here. Last, but certainly not least: Khin Khin Lay. She invested an incredible amount of time in this project, even though when she first started she found herself committed to a year of research on a project she didn't choose and didn't seem to have the training for. I'll consider myself extremely lucky if I ever again work with someone who could so cheerfully face the prospect of repeating the same ten hours of tedious experiments, again, and again, and again, when it seemed like all we were doing was throwing away all the protein she'd invested weeks in making. Much of the cloning and expression work is the product of her hands, as is a large fraction of the ligand binding and thermal denaturation data. The fact that we have reliable binding data at all is due to her persistence and willingness to perform all the bizarre experimental variations I could think of. Although I'm sure she doesn't believe it, this project would still be in a complete disarray without her efforts.

My extended family, and especially my sisters and parents, have my thanks for all their support and understanding that graduate school is somewhat unpredictable.

Finally, I thank my wonderful wife, Dr. Shannon Watt, without whom I'd be a much poorer person. I cannot imagine someone it would have been better to journey these last few years with.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

9cRA	9-*cis* retinoic acid
A	Alanine (amino acid context) Adenine (DNA/RNA context)
AB	A/B Domain (nuclear receptor context)
AD	Activation domain
AF-2	Activation function domain 2
AMAT	hRXRα(D_{β}E)-I268A;I310M;F313A;L436T
APAF1	Apoptotic peptidase activating factor 1
ASAF	hRXRα(D_{β}E)-I268A;I310S;F313A;L436F
atRA	all-*trans* retinoic acid
AUC	Analytical ultracentrifugation
B	Any of: aspartate, asparagine (amino acid context) Any of: cytosine, guanine, thymine, uracil (DNA/RNA context)
BAX	BCL2-associated X protein
BLAST	Basic Local Alignment and Search Tool
bp	Base pairs
C	Cysteine (amino acid context) Cytosine (DNA/RNA context) DNA binding domain (nuclear receptor context)
CD	Circular dichroism
cm	Centimeter
CMI	hRXRα(D_{β}E)-Q275C;I310M;F313I
Cys	Cysteine
D	Aspartate (amino acid context) Any of: adenine, guanine, thymine, uracil (DNA/RNA context) Dimeric state (equilibrium context) Hinge domain (nuclear receptor context)
D_{α}	RXR residues 200-211 (first half of hinge domain)
D_{β}	RXR residues 212-224 (second half of hinge domain)
Da	Dalton (atomic mass unit)
DBD	DNA binding domain
dH_2O	Deionized water

DNA	Deoxyribonucleic acid
DSC	Differential scanning calorimetry
E	Glutamate (amino acid context) Ligand binding domain (nuclear receptor context)
E. coli	*Escherichia coli*
EC_{50}	Effective concentration (ligand concentration at 50% efficacy)
Eff.	Efficacy
ESPSearch	Exact Sequence and Pattern Search
F	Phenylalanine
G	Glycine (amino acid context) Guanine (DNA/RNA context)
GBD	GAL4 DNA binding domain
GdmCl	Guanidinium chloride
GST	Glutathione-s-transferase
h	Human
H	Histidine (amino acid context) Any of: adenine, cytosine, thymine (DNA/RNA context)
His	Histidine
I	Isoleucine (amino acid context)
ICAT	Isotope-coded affinity tag
IGF-BP3	Insulin-like growth factor binding protein 3
IL	hRXRα($D_{\beta}E$)-F313I;F439L
IPTG	Isopropyl-β-D-thiogalactopyranoside
IUPAC	International Union of Pure and Applied Chemistry
J	Any of: isoleucine, leucine, valine
K	Lysine (amino acid context) Any of: guanine, thymine, uracil (DNA/RNA context) Kelvin (unit context)
K_d	Dissociation constant
kDa	Kilodalton
kJ	Kilojoule
L	Leucine (amino acid context) Liter (unit context)
LB	Luria-Bertani

LBD	Ligand binding domain
LMM	hRXRα(D_{β}E)-I310L;F313M;L436M
LxxLL	C-^{688}steroid receptor coactivator-1^{702}
µg	Microgram
µM	Micromolar
m	Mouse
M	Methionine (amino acid context) Any of: adenine, cytosine (DNA/RNA context) Monomeric State (equilibrium context) Molar (unit context)
mg	Milligram
min	Minute
mL	Milliliter
mM	Millimolar
mol	Mole
MPAX	Misincorporation proton-alkyl exchange
mRNA	Messenger RNA
N	Asparagine (amino acid context) Any DNA or RNA base (DNA/RNA context)
N/A	Not applicable
NaP_i	Sodium phosphate
NCBI	National Center for Biotechnology Information
N.D.	No data
ng	Nanogram
nM	Nanomolar
nm	Nanometer
NMR	Nuclear magnetic resonance spectroscopy
OD	Optical density
OPBA	γ-oxo-1-pyrenebutyric acid
P	Proline
p53AIP1	p53-regulated apoptosis-inducing protein 1
PAGE	Polyacrylamide gel electrophoresis
PCR	Polymerase chain reaction

PPAR	Peroxisome proliferator-activated receptor
PTEN	Phosphatase and tensin homolog
Q	Glutamine
R	Arginine (amino acid context) Any of: adenine, guanine (DNA/RNA context)
RAM	Random access memory
Ref.	Reference
Resp. Elem.	Response element
RNA	Ribonucleic acid
rpm	Revolutions per minute
RXR	Retinoid X receptor
S	Serine (amino acid context) Any of: cytosine, guanine (DNA/RNA context)
SDS	Sodium dodecyl sulfate
sec	Second
siRNA	Small interfering RNA
Sp.	Species
SVM	hRXRα(DβE)-A272S;I310V;F313M
T	Threonine (amino acid context) Thymine (DNA/RNA context) Tetrameric state (equilibrium context)
TLT	hRXRα(DβE)-A272T;I310;F313T
T_m	Melting temperature
U	Any of: serine, threonine (amino acid context) Uracil (DNA/RNA context) Unfolded state (equilibrium context)
UV	Ultraviolet
UV-vis	Ultraviolet-visible
V	Valine (amino acid context) Any of: adenine, cytosine, guanine (DNA/RNA context)
VVMSM	hRXRα(DβE)-I268V;A272V;I310M;F313S;L436M
VVLM	hRXRα(DβE)-I268V;A272V;I310L;F313M
W	Tryptophan (amino acid context) Any of: adenine, thymine, uracil (DNA/RNA context)
wt	Wild-type

X	Any amino acid
xg	Times gravitational force
Y	Tyrosine (amino acid context) Any of: cytosine, thymine, uracil (DNA/RNA context)
Z	Any of: glutamate, glutamine

SUMMARY

The human retinoid X receptor α (hRXRα) is a member of the nuclear receptor super-family of ligand-activated transcription factors. The Doyle laboratory has previously engineered a variety of functional hRXRα variants that activate gene expression in response to synthetic ligands (LG335 and γ-oxo-1-pyrenebutyric acid), compounds that are poor activators of wild-type hRXRα. The variants generally no longer respond to the wild-type ligand 9-*cis* retinoic acid.

To enable targeting of these engineered receptors to arbitrary DNA sequences, we developed a program, ESPSearch, for identifying short or specific sequences in DNA or protein. ESPSearch enables identification of combinations of known zinc finger motifs to target arbitrary genes, as well having several other applications. The ability to target any DNA sequence means that the engineered receptors can be directed to control any gene.

The ligand binding, self-association, coactivator interactions, and unfolding properties of the ligand binding domain of wild-type hRXRα were characterized. Our expression and purification protocol improves upon existing methods, providing high purity protein in a single step with more than twice prior yields. A general fluorescence-based method for measuring ligand affinity with hRXRα was developed, and used to determine binding constants for the small molecules. The presence of a peptide containing the binding motif from coactivator proteins (LxxLL) differentially increased the affinity of the receptor for the ligands. Assays to determine the self-association give a K_d for the dimer-tetramer equilibrium of 35 μM. hRXRα was found to denature irreversibly when heated, but shifts in apparent T_m due to ligands correlates strongly with the ligand binding affinities. Our results clarify disparities in existing reports and provide a benchmark for

comparison. Reliable analysis of our data led to the development of a computer program for rigorous, automated data fitting.

Nine functional variants of hRXRα were characterized to probe correlations between biophysical properties and the observed functional activity of the receptors, which differ significantly from wild-type. Although the correlation between ligand binding affinity and melting temperature was strong for all variants, there was essentially no correlation between ligand binding and activation of the variants. The mutations, which are all contained within the binding pocket, have significant long-range effects on the protein, causing changes in ligand-LxxLL interactions and oligomerization of the variants. Experimental and computational analysis of selected mutations suggests that they are highly coupled, complicating protein design. However, the large variation in properties amongst the variants also suggests that hRXRα can be mutated extensively while still retaining function. The long-range impact of binding pocket mutations will need to be taken into account in future engineering projects, as hRXRα is a flexible, dynamic protein.

CHAPTER 1
RETINOID X RECEPTORS

1.1 Overview of Nuclear Receptors

The retinoid X receptors (RXRs) are members of the nuclear receptor super-family of ligand-activated transcription factors (1). Nuclear receptors are found only in animals and plants, although recent evidence has suggested that fungi may share a nuclear receptor-like ancestral protein with the nuclear receptor superfamily (2). Current evidence suggests that the first nuclear receptor was most similar to the present-day estrogen receptor, and that other nuclear receptors evolved later (3). The nuclear receptor superfamily is composed of several groups, and a rational nomenclature based on sequence alignments and a constructed phylogeny has been proposed to name receptors based on their subfamily and gene(s) (4, 5). One of the largest subfamilies is the classical steroid receptors, while includes the estrogen receptors (subfamily NR3A), the androgen receptor (subfamily NR3C, gene C4), the glucocorticoid receptor (subfamily NR3C, gene C1), and the mineralocorticoid receptor (subfamily NR3C, gene C2). Another large subfamily is composed of receptors that tend to heterodimerize with RXRs (see below), including the pregnane X receptor (subfamily NR1I, gene I2), the retinoic acid receptors (subfamily NR1B), the peroxisome proliferator-activated receptors (subfamily NR1C), the vitamin D receptor (subfamily NR1I, gene I1), the thyroid receptors (subfamily NR1A), the ecdysone receptor (subfamily NR1H, gene H1), and the retinoid-related orphan receptors (subfamily NR1F). The retinoid X receptors (subfamily NR2B) are part of a much smaller group of related receptors.

As transcription factors, nuclear receptors regulate gene expression, and are important in a wide variety of biological processes, including steroid

signaling, metabolism, and xenobiotic sensing (5, 6). A single nuclear receptor is likely to be involved in numerous pathways within the cell, and many nuclear receptors have been shown to be involved in pathways that do not directly involve gene regulation (5).

Nuclear receptors exert their gene-regulation by binding to specific DNA sequences in gene promoters called response elements (7, 8). These response elements are generally composed of six specific DNA bases; which six varies somewhat across the nuclear receptor groups. For example, the androgen receptor binds the DNA sequence AGAACA, while RXRs instead recognizes AGGTCA (7). In addition, many nuclear receptors bind groups of closely related DNA sequences or are sensitive to DNA bases outside the six being directly bound (9-18). Although a few nuclear receptors can bind to DNA and regulate gene expression as a single, monomeric unit, most require another nuclear receptor as a binding partner (7, 8). In some cases, such as the estrogen receptors and RXRs, the partner may be another identical protein, and hence they form homodimers, while other receptors require a different nuclear receptors as partner to form a heterodimer (7). A small number of nuclear receptors are able to form higher order units: the thyroid receptor has been reported to form trimers (19), while RXRs can form tetramers (20, 21).

In addition to DNA and each other, nuclear receptors bind coactivator and corepressor proteins, which respectively help activate and suppress activity of the nuclear receptors (5). The coactivator proteins contain specific motifs, generally a helical LxxLL amino acid sequence (where x designates any amino acid), which bind to the surface of the nuclear receptors and stabilize active conformations (5, 22). Some nuclear receptors have also been reported to interact directly with members of additional protein families (23, 24).

Nuclear receptors can also be regulated by post-translational modifications, including phosphorylation, sumoylation, and ubiquitination. Specific effects of these modifications are different for each receptor; the transcription activity of RXRs is generally reduced by these mechanisms. Phosphorylation can upregulate the activity of RXR-retinoic acid receptor heterodimers (25), or deactivate RXRs by reducing DNA binding affinity (26-31) and triggering export from the nucleus (32). Sumoylation has been shown to deactivate RXRs (33).

Most nuclear receptors can be regulated by transport mechanisms. As transcription factors, nuclear receptors are inactive unless they are within the nucleus of the cell. Chaperone proteins, in response to some other signal, have been identified as transporters of nuclear receptors across the nuclear membrane (34, 35). Some evidence for shuttling of RXRs exists (36), but other reports suggest that RXRs exist predominantly in the nucleus unless specifically exported (37, 38). Nuclear receptors not present in the nucleus may still be involved in other roles not related to regulation of transcription (5).

One method of nuclear receptor control that is of particular interest to the medical community is regulation via small molecule binding; it is this interest which caused the rapid growth of publications and maintained interest in nuclear receptors such as RXRs (see Figure 1.1). Nuclear receptors undergo a conformational change to an active state upon binding agonist ligands, while antagonist ligands tend to block the conformational change (5, 39-41). An extensive vocabulary now exists to classify the precise effect of ligands on receptors, which takes into account factors such as how much additional factors (e.g., cell-type specific effects) influence the degree of activation by an agonist (5). For our purposes, small molecules that bind a receptor a ligands; those that also happen to cause some degree of

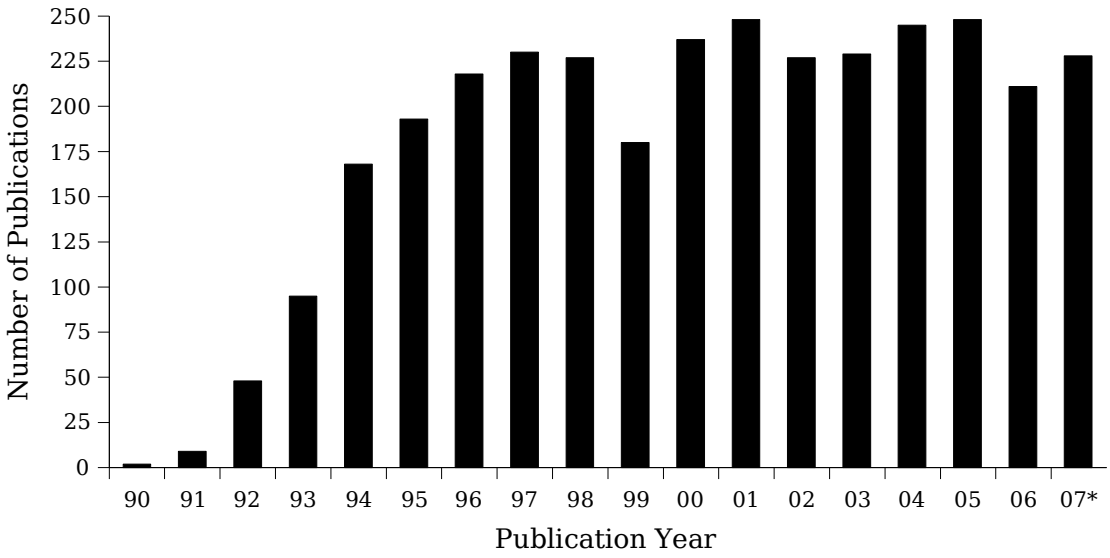

Figure 1.1. History of RXR-related publications. Bars represent the number of publications mentioning "RXR," "retinoid X receptor," or "ultraspiracle AND receptor" in the abstract of PubMed indexed journals. *=projected based on first 9 months. The few cases "RXR" having other meanings were removed.

activation are called agonists.

1.2 Function and Structure of RXRs

Like most nuclear receptors, the retinoid X receptors are composed of four primary domains (see Figure 1.2). The N-terminal domain is designated the A/B domain, and is the most variable domain between different nuclear receptors; some nuclear receptors lack this domain entirely or have a domain much larger than the A/B domain in RXRs (5, 42). This domain contains the activation-function domain 1, which plays a role in binding DNA and coactivator protein interactions (43-46). The next domain is the DNA-binding domain (DBD), designated C, which is composed of two Cys_4 zinc finger domains (47). This domain is the most highly conserved domain, and is the primary domain used to classify receptors into subfamilies, although a very few nuclear receptors lack this domain (4, 5). Next comes the hinge, or D, domain, which is a variable linker region, although recent reports have

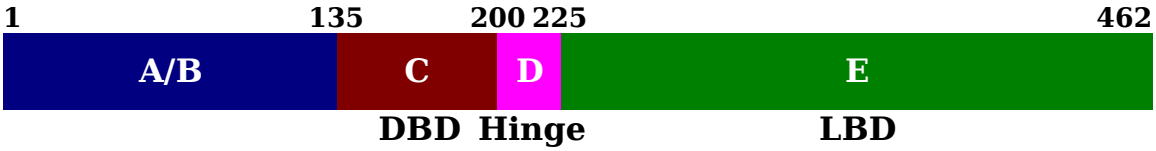

Figure 1.2. hRXRα domains. Numbers indicate the amino acid position at the beginning or end of each domain.

suggested that the domain is functionally important (48-50). Finally, the ligand binding domain (LBD), or E, contains the ligand binding pocket and activation-function domain 2 (AF-2) (5, 42). AF-2, which contains the final α-helix of RXRs, folds over the binding pocket when ligands activate the receptor.

As is common to nuclear receptors, multiple subtypes[1] of RXRs exist in vertebrates, designated α, β, and γ (genes B1, B2, and B3, respectively) (42, 51). Although the three subtypes are separate genes and have considerable variation, especially in the A/B domain, they are very closely related from a functional and practical standpoint; no ligand has yet been found that discriminates between the three subtypes to any appreciable extent (42). The bulk of the publications to date focus on RXRα or make no distinction between subtypes; we use the α-subtype exclusively in the work described below. Each subtype also exists as two isoforms that differ somewhat in the A/B domain, but no significant work has yet been performed on the importance of these isoforms (42). In addition to all these variations, the invertebrate homolog is designated the ultraspiracle receptor (USP, subfamily NR2B, gene B4) (52).

The domain codes, in addition to species and subtype, will be used throughout this manuscript to specify precisely which RXR is being discussed. For example, hRXRα(CDE) refers to human RXRα lacking the A/B

1 Use of the words "subtype" and "isoform" conform to the standard nuclear receptor nomeclature (4).

domain, while mRXRα(E) refers to the mouse RXRα ligand binding domain. In addition, the D (hinge) domain may also be designated as D_{α} or D_{β} to refer to only the first or second half of the domain, respectively, so that hRXRα(D_{β}E) refers to the second half of the hinge and all the LBD (amino acids 212-462) of human RXRα.

The retinoid X receptors serve a variety of roles in animals (53). Inactivation of RXRα leads to the death of mice during embryonic development (54, 55). Only half of RXRβ knockout mice died before birth, but the survivors were sterile (56); another report suggests that only RXRα is essential, as it can fill the roles of RXRβ and RXRγ (57). The three subtypes have somewhat different expression patterns in cells, which leads to some differences in function simply due to differing concentrations in specific cell types (42). In addition to their roles as a heterodimer partner and sometimes regulatory element for numerous other nuclear receptors, RXRs (generally RXRα) have been found to be have direct roles in differentiation (51, 58-84), metabolism and diabetes (85-97), and the immune system and autoimmune diseases (98-122). RXRγ may have specialized roles in formation of memories (123) and metabolic processes (124, 125). Agonists for RXRs are currently a focus of much medical research for cancer treatments (126-168), and RXRs have a role in the progression of some viral diseases (169-175).

The first agonist identified for RXRs, 9-*cis* retinoic acid (9cRA) (176, 177), is still the most potent naturally occurring ligand. While it has generally been assumed to be the biologically relevant agonist for RXRs, a significant body of work has carefully examined the question, and concluded that this assumption is incorrect (178-181). Other ligands that may instead naturally activate RXRs include docosahexanoic acid and other fatty acids (178, 182-187).

No crystal structure for a full-length nuclear receptor yet exists. The report closest to this goal is a low-resolution structure of hRXRα(CDE) (188). hRXRα(E) was the first nuclear receptor ligand binding domain to be crystallized (189). Since this initial report, RXRα(E) (from several species) has been crystallized many times, both alone (190-193) and as part of a heterodimer (194-200). In addition, an NMR solution structure of hRXRα(E) has been reported (201). Crystal structures have also been reported for RXRβ(E) (202-204) and USP(E) (205-209). As it was the first LBD structure solved, RXRα is frequently used as a general model for descriptions of nuclear receptors. The ligand binding domain of RXRα is predominantly made up of α-helices, with only two short β-strands; although the exact composition varies within the nuclear receptor superfamily, the general fold and dominance of α-helices is common to all known ligand binding domains (5). In addition to these structures, several NMR solution structures have been reported for the hRXRα DBD (210-214).

1.3 Engineering RXRα

Nuclear receptors are modular, meaning that the natural DNA binding domain can be replaced with a different DBD, leaving an intact and functional LBD (5). Because it is already activated by small molecules and can be made to regulate an essential gene, RXRs (and nuclear receptors in general) are attractive targets for protein engineering, the numerous applications of which have recently been reviewed (215).

Previous work in the Doyle laboratory (216-221) has used a variety of techniques to engineer orthogonal ligand-receptor pairs (222). Orthogonal receptors are protein variants that are inactive in the presence of the wild-type ligand, but respond to a ligand that does not activate the wild-type receptor (see Figure 1.3). Methods used to create receptors with this functionality include site-directed mutagenesis (216, 217) and genetic

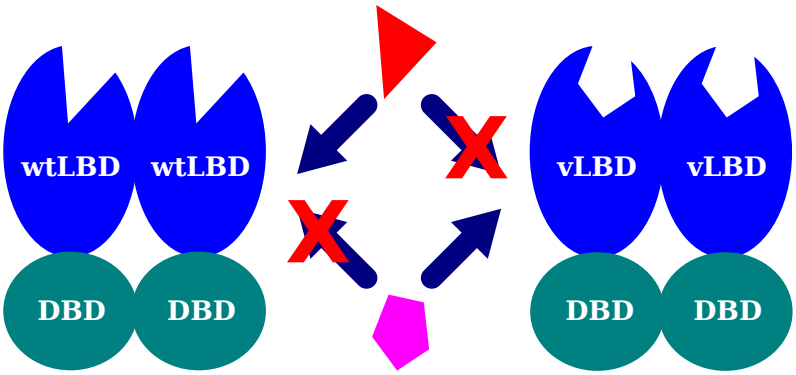

Figure 1.3. Orthogonal ligand-receptor pairs. Wild-type receptors respond to the wild-type ligand (red triangle, ▶), but not the synthetic ligand (magenta hexagon, ⬢). The variant receptors, which contain mutations in the LBD, respond to the synthetic ligand, but not to the wild-type ligand.

selection of receptors from libraries constructed using structure-guided codon randomization (219-221). The genetic selection technique, chemical complementation, was developed in the Doyle laboratory and is depicted in Figure 1.4. Briefly, yeast lacking an essential gene are transformed with a plasmid containing the essential gene with a specific promoter (the DNA in Figure 1.4). A second plasmid contains a coactivator protein fused to the GAL4 activation domain. A third plasmid contains the gene for an engineered hRXRα(DE) fused to the GAL4 DNA binding domain. The promoter chosen to regulate the essential gene is one that is activated by GAL4, a modular (ligand-independent) yeast transcription factor. The result is a system whereby yeast transcription machinery only "sees" yeast proteins, but the assembly of the "complete" GAL4 transcription factor requires that the two human proteins bind together. For this to occur, a ligand that activates hRXRα must be present in the system. Hence, when a large library of hRXRα variants is transformed at once, genetic selection experiments on the library in the presence of the desired target ligand will result in survival of only the yeast containing a receptor activated by the ligand.

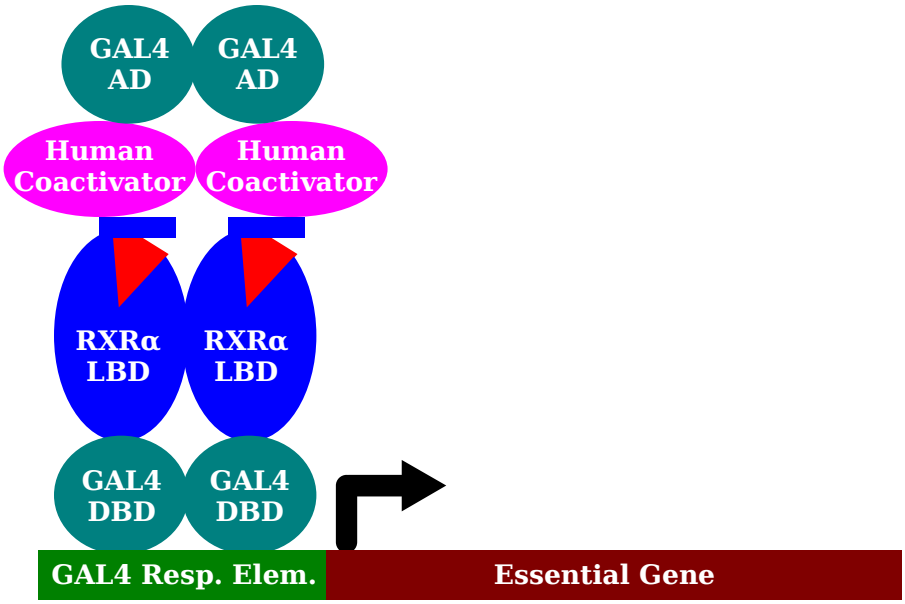

Figure 1.4. Schematic of chemical complementation. hRXRα(DE) (blue ellipse, ●) is fused to the GAL4 DBD (turquoise circle at bottom, ●) while a human coactivator (magenta ellipse, ●) is fused to the GAL4 activation domain (turquoise circle at top, ●). Human proteins interact with each other in the presence of an activating ligand (red triangle, ▼), while only GAL4 is exposed to other proteins in the cell and the DNA response element (green rectangle, ▬). When the ligand activates the receptor and recruits the coactivator-GAL4 AD fusion protein, the essential gene (dark red rectangle, ▬) is transcribed.

Two (of many) potential applications of these engineered variants are shown in Figures 1.5 and 1.6. In the first example, a gene used in gene therapy, perhaps to supply a protein that is otherwise missing and therefore causing a disease, is delivered into the patient as part of a larger construct containing an engineered hRXRα and a response element for the engineered receptor. Unlike traditional gene therapy, where dosing is controlled entirely by the quantity of gene injected and hence is difficult to regulate, this system instead constitutively expresses the orthogonal hRXRα variant. Unless an oral drug is administered, this engineered variant is inactive (as it is selected to not respond to the small molecules naturally present in the body), and the therapeutic gene is not transcribed (Figure 1.5, top). This scheme allows for more precise control of the therapeutic protein levels

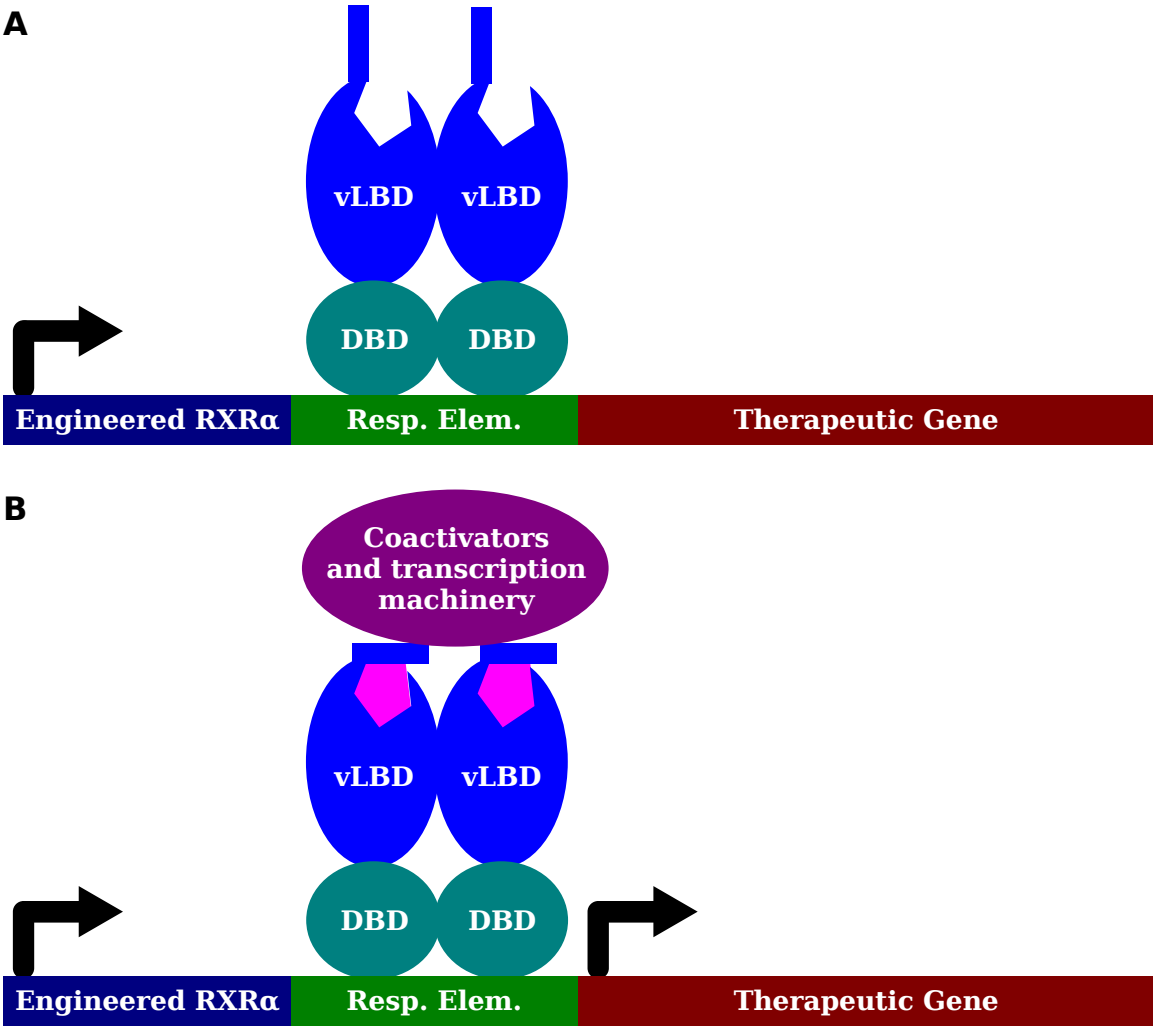

Figure 1.5. Gene therapy using an engineered receptor. (A) Only the engineered RXRα gene (blue rectangle, ▬) is transcribed in the absence of drug, not the therapeutic gene (red rectangle, ▬). The engineered receptor remains in the inactive conformation. (B) When the drug (magenta hexagon, ⬢) is delivered and binds to the receptor, the engineered RXRα undergoes a conformational change and the therapeutic gene is transcribed.

ultimately present in the patient. A similar system could be imagined for agricultural use, where crops are engineered to express pesticides only when a specific trigger activates the nuclear receptor, such as an otherwise benign chemical sprayed on the crops, or a compound present only when crops are being actively attacked by pests.

A second application is in building sensors. The first step is to

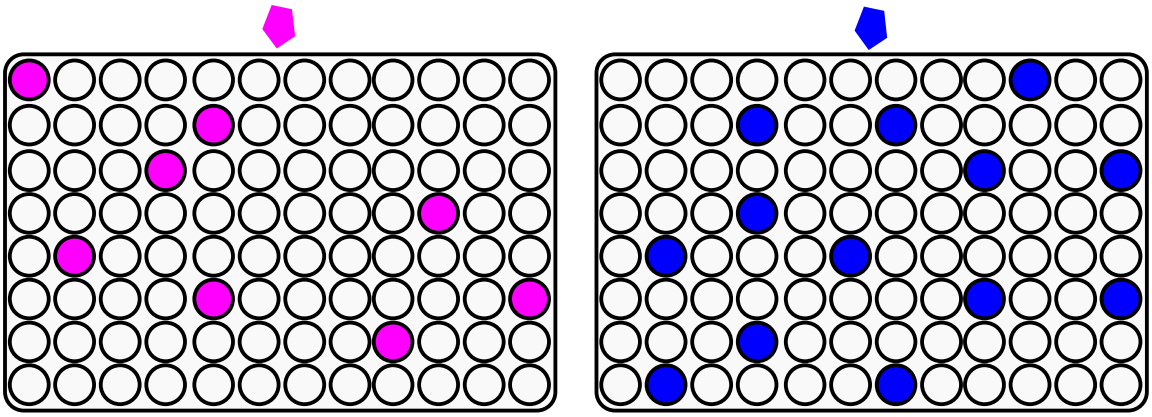

Figure 1.6. Sensor array using engineered RXRα variants. Each well contains a single variant receptor. One ligand activates a selection of receptors (left), while a second activates a partially-overlapping second set (right). The pattern associated with each ligand serves as a "fingerprint."

engineer many receptors, each with a unique profile in response to a variety of chemicals, but not necessarily specific for any one. Next, the variants are coupled to some output triggered by the active conformation of the receptor. Finally, the variants are individually placed into wells on a plate, to create a general sensor that is able to "fingerprint" potentially thousands of compounds. By coupling activation of the receptor to a reaction that produces visible light, a system such analagous to shown in Figure 1.6 could be constructed, where two compounds that share one property (shape) but differ in another (color) yield distinct patterns.

Both site-directed mutagenesis and chemical complementation have produced receptors with varying degrees of activity and selectivity toward 9-*cis* retinoic acid and synthetic ligands LG335 and γ-oxo-1-pyrenebutyric acid (OPBA). Although some early variants were characterized for ligand affinity (216), most of the variants have only been examined from a functional standpoint. The specific causes for the range in selectivities, maximal activities, and minimal ligand concentrations necessary for survival have not been apparent from functional and computational analyses. The characterization of several variants, as well as a more rigorous

characterization of wild-type hRXRα than has previously been undertaken, is the focus of most of the work that follows. The remainder of the work focuses on other aspects of the engineering problem, including how to target the variants to unique sites in the genome and rigorous data analysis methods.

CHAPTER 2
ESPSEARCH: A PROGRAM FOR FINDING EXACT SEQUENCES AND PATTERNS IN DNA, RNA, OR PROTEIN

2.1 Introduction

The availability of protein and genomic databases makes identifying specific sequences from within long sequences and databases a common activity, such as for small interfering RNAs (siRNAs), artificial transcription factor binding sites, and protein motifs. However, currently available tools tend to be highly specialized and suited for only a small number of applications. In addition, no general tool exists for identifying patterns made of up several other sequences when those sequences can be arranged in a large number of ways, such as identifying specific or unique DNA binding sites for a protein constructed from a pool of zinc finger domains. We are interested in this problem as a method of specifically targeting an engineered nuclear receptor to an arbitrary, specific gene, but the problem is the same regardless of context.

We designed a computer program, ESPSearch (*E*xact *S*equence and *P*attern *Search*), to address this lack. ESPSearch was designed with five specific requirements in mind. First, the program must allow the user to specify exactly which target sequences are being searched for, and allow for several sequences at once. Most current tools fail this requirement, especially fixed databases of predetermined target sequences, such as the TRANSFAC® (223) transcription factor database tools (only transcription factors) and tools for identifying restriction enzyme recognition sites, as well as BLAST (224) and the PROSITE (225) database tools that can only search

---

Material from this chapter has previously been published as Watt, T. J. and Doyle, D. F. (2005) ESPSearch: A Program for Finding Exact Sequences and Patterns in DNA, RNA, or Protein, *Biotechniques 38*, 109-115. Reprinted with permission.

for a single sequence at a time. Moreover, while BLAST is excellent at identifying related sequences of sufficient length, it makes several assumptions useful for some applications but poorly suited for identifying shorter sequences with particular constraints; one example is the use of a "word size" to increase search speed by forcing a certain number of consecutive exact matches between the sequence being searched for and the sequence being searched in.

Second, the program must be able to identify target sequences of any length, and with possibly highly variable composition (i.e., the targets may contain wildcards or built-in gaps). Some PROSITE (225) and European Molecular Biology Open Software Suite (226) tools also have this capability, but only for a single sequence at a time.

Third, the program must be able to match a pattern of the identified target sequences meeting arbitrary (user-specified) criteria, through an optional, separate logical step. Although many existing tools search for "patterns," these are equivalent to what we define as a "target sequence," which is a single sequence that may contain wildcards, specific gaps, and mismatches. When we refer to "patterns," we mean a more complex sequence consisting of groups of individual target sequences. For example, a simple pattern is the binding site of a heterodimeric protein on DNA, which can be represented as AB, where each letter represents any target sequence from a database; the pattern corresponds to the entire heterodimer binding site. The web application PatSearch (227) is a powerful tool based on regular expressions for matching patterns, but it only searches for a single pattern at a time and must refer explicitly to subunits (target sequences), so complex searches with PatSearch might require millions of individual searches. A tool to search all these combinations in one step does not exist elsewhere.

Fourth, the program should have an intuitive input format that does not require knowledge of regular expression formatting. Most general tools currently available are based on the use of regular expressions (such as in PROSITE and PatSearch), which are powerful but difficult to learn and understand. In addition, programs that rely on regular expression formatting tend to avoid searching for multiple, distinct sequences at once. Instead, search utilities tend to focus on how to represent all target sequences with a single regular expression, which can be difficult or impossible for some searches, meaning that multiple searches must be performed serially.

Finally, the program must be free and available for modification. Many excellent search tools are commercially available, but are expensive and frequently allow little or no user-control over most parameters and assumptions. A general-purpose tool should not only be widely available, including running on many different operating systems, but also should have all search parameters and the source code available for modification.

We demonstrate some applications of ESPSearch by designing a variety of possible artificial transcription factors, estimating the specificity of artificial transcription factors and siRNAs, and identifying possible protein-protein binding sites.

2.2 Program Description

ESPSearch was written in Python, a scripting language available for nearly any operating system, and should run on any computer with Python installed. Our searches were performed on a PC with a 1.8 GHz Pentium® 4 processor and 1 GB of RAM running Microsoft® Windows® 2000, but ESPSearch has also been tested on various other hardware and operating systems. All searches were performed under non-stringent conditions (i.e., background applications such as antivirus software and firewall present with

a live network connection). ESPSearch, instructions, and examples (including the files used for the examples below) may be downloaded at http://web.chemistry.gatech.edu/~doyle/espsearch/ (and in the Appendices). Also available is a graphical user interface for setting up runs.

ESPSearch loads a user-specified database of target sequences (the sequences to be searched for). All variations of target sequences due to wildcards and mismatches are expanded, and each sequence is assigned a unique memory address. For nucleic acid searches, reverse complements to each sequence are also generated to identify matches on the complement strand without actually generating or searching the complement. ESPSearch loads a specified rule set that allows it to interpret the target sequences and source sequence, and how they relate to each other (e.g., that A and T are complements, or that X represents any amino acid). The source sequence (the sequence to be searched) is loaded in steps into a buffer, by default approximately 1,000 positions long. The program automatically determines and uses a fixed "window" length to examine the sequence in the buffer, stepping through it to examine each possible window. The window size is automatically set, and varies with the specifics of each database. The memory address that corresponds to each window is generated, and that address is checked to determine if one or more target sequences match: if the address does not exist, then no sequences match; if it does exist, the matching sequences are output in one of several available formats. Databases containing target sequences with gaps or with a variable length use a fixed window to determine if the first part of the sequence matches, and then searches forward in the buffer to determine if the next positions correspond to the remaining portions of the tentatively identified target sequences. Upon output, the location, sequence, and matching target sequence name are stored for pattern processing.

Upon completing the scan of a buffer for individual target sequences, pattern processing is performed on the identified sequences. Each located target sequence is sequentially assigned to the first position in each pattern. The following positions are then checked for target sequences that match the remaining pattern positions at the specified relative distance (gaps may be specified at particular locations in patterns). When evaluating whether a target sequence is acceptable in the pattern, the following variables may be specified: the strand the target sequence occurs on relative to the first position in the pattern (for nucleic acid searches); whether the position is marked as required to be unique (i.e., the target sequence found at that position must not be found anywhere else in the pattern); whether the target sequence in the position needs to be identical to some other position in the pattern; and for positions marked as identical, whether the sequences are physically identical (i.e., the same sequence) or simply recognized by the same target(s), which is an important distinction when wildcards and mismatches are used. Upon completing the scan of a buffer for all patterns, a new buffer is loaded, retaining the necessary information from the previous buffer to ensure that patterns that occur across buffer loads are not missed. The patterns used here are represented in a slightly simplified form, where uppercase and lowercase letters represent target sequences found on different strands (e.g., Aa represents two sequential target sequences, but one on each strand of the source sequence) and gaps are represented as a subscripted numerical range.

The speed of a search varies linearly with the length of the source sequence. A typical search of the *Saccharomyces cerevisiae* genome, 12.1 million base pairs, takes 4-12 seconds. In addition, the amount of output generated by a search also affects the speed; when the output generated (in bytes) is less than the source sequence, this factor is relatively unimportant.

The complexity of the target sequences also influences search speed, but only tremendously complex target sequences (e.g., multiple gaps with very large ranges in each target sequence, many wildcards, and numerous mismatches tolerated) have a significant effect. Memory usage is dependent on the total number of expanded sequences to be identified, but only exceptional searches use more than a few megabytes of RAM.

2.3 Applications

2.3.1 Identification of Heterodimeric Artificial Transcription Factors

ESPSearch was used to design possible dimeric artificial transcription factors that utilize human zinc fingers to bind DNA. The artificial transcription factors were designed to target p53 binding sites, because p53-regulated genes involved in apoptosis are frequently turned off in cancerous cells due to mutations in p53 (228). Zinc fingers are common, modular peptide units that recognize three base pair (triplet) DNA sequences and can be fused in a single protein to recognize longer sequences and activate genes (229, 230). Reactivation of a p53-regulated gene using a zinc finger-based artificial transcription factor has recently been demonstrated (231).

To identify possible artificial transcription factors, we created a database containing the DNA triplet(s) reported to be recognized by 56 human zinc fingers, which include several wildcard positions (232), and used a rule set corresponding to standard IUPAC notation for DNA. A dimeric construct was used for four reasons: dimeric proteins tend to be more specific (more likely to recognize a sequence that is unique) than monomeric proteins (233-235); most artificial transcription factors to date are monomeric, but an example of a dimeric protein based on engineered zinc fingers was recently reported (236); nuclear receptors, and RXRα in

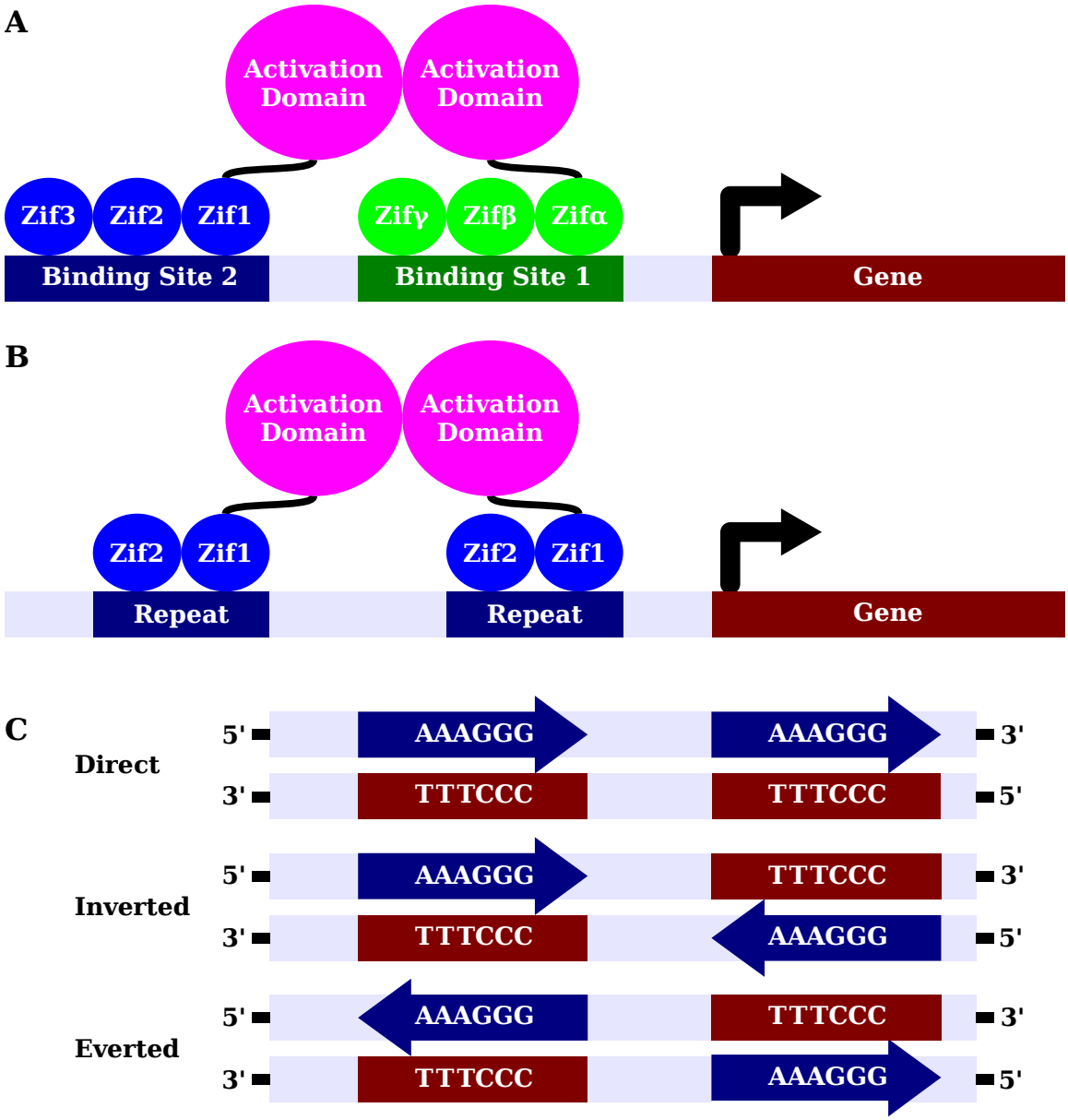

Figure 2.1. Artificial transcription factor design using zinc fingers. (A) Heterodimer design. (B) Homodimer design. (C) Repeat patterns used to find homodimer binding sites: direct (ABAB), inverted (ABba), and everted (abBA).

particular, generally function as dimers; and because a dimeric system is a more challenging design problem that illustrates the complexity ESPSearch can address.

We searched for binding sites in 12 known human p53 recognition

sites (in nine p53-regulated genes) (237-241). The majority of the p53 recognition sites are fewer than 25 base pairs long. The total number of base pairs to be recognized is 18; therefore each monomer needs to contain three zinc fingers (Figure 2.1A). A variable gap of 0-6 base pairs is allowed between the binding sites for each monomer. The pattern corresponding to these constraints is $ABC_{0-6}DEF$: any three zinc finger binding sites, a gap of up to six base pairs, and any three additional zinc finger binding sites. Each group of three triplets could be on either strand, so we also used the pattern $ABC_{0-6}def$.

ESPSearch scans the source sequence for sequences matching the desired patterns. The matches are the number of possible binding sites for, and therefore the specificity of, each pattern, based on the selectivity for DNA of the zinc fingers that comprise the pattern. Of the twelve p53 recognition sites searched, ESPSearch identified sequences matching the pattern in six (regulating five genes); the identified sequences and corresponding zinc finger proteins are shown in Table 2.1. Several p53 recognition sites contain more than one possible heterodimer (e.g., *BAX*). In addition, several human zinc fingers bind the same triplet, so there are often several choices for a zinc finger at a given position within a monomer. The specific zinc fingers shown in Table 2.1 were chosen on the basis of specificity, and arbitrarily in the event of multiple zinc fingers with comparable specificity. Binding affinity or selectivity of individual zinc fingers could be used to further refine choices, but that information has not yet been reported for all of the zinc fingers.

Each of the six sites where a heterodimer was not identified contain at least one set of three sequential recognized triplets (i.e., half a heterodimer). By extending the source sequence three additional base pairs 5' and 3' beyond the p53 recognition site in the search, heterodimers for two

Table 2.1. Heterodimers and recognized sequences in p53 recognition sites matching the pattern $ABC_{0-6}DEF$ or $ABC_{0-6}def$.

		Identified Zinc Fingers for Heterodimers[α] (Recognized Sequence[β])	
Gene (reference)	**p53 site (bp[γ])**	**First Protein**	**Second Protein**
APAF1 (237)	27	VSTR;DSAR;VSTR (GCWRTCGCW	RDER1;HSNK;RSHR GGGRACGHG)
APAF1 (237)	33	QSHR3;RDHT;DSAR (RTCNGGRGA N_5	QSNI;QSHT;QSHR2 GGAHGAMAA)
BAX (238)	39	DSCR;QSNI;QSHR3 (RGAMAAGCC	RDHT;RDER2;RDHT NGGGCGNGG)
		RDHT;DSCR;QSNI (MAAGCCNGG	VSTR;RDHT;RDER2 GCGNGGGCW)
		QSHR3;HSSR;QSNI (MAAGTTRGA N_2	RDHT;DSCR;QSNI MAAGCCNGG)
		HSNK;QSHR3;HSSR (GTTRGARAC N_2	RDER2;RDHT;DSCR GCCNGGGCG)
		DSCR;QSNI;QSHR3 (RGAMAAGCC N_3	VSTR;RDHT;RDER2 GCGNGGGCW)
		QSHR3;HSSR;QSNI (MAAGTTRGA N_5	RDER2;RDHT;DSCR GCCNGGGCG)
		HSNK;QSHR3;HSSR (GTTRGARAC N_5	RDHT;RDER2;RDHT NGGGCGNGG)
IGF-BP3 (239)	152	68 pairs (not shown)	
p53AIP1 (240)	20	DSCR;QSNI;QSHT (HGAMAAGCC	QSHR3;QSSR1;RDHT NGGGCWRGA)
		DSCR;QSNI;QSHT (HGAMAAGCC N	RSNR;QSNI;RSHR GGGMAAGAG)
PTEN (241)	34	DSCR;QSNI;RSNR (GAGMAAGCC	VSTR;DSCR;RDHT CCNGGCWGC)
		DSCR;QSNI;RSNR (GAGMAAGCC N_3	QSSR1;VSTR;DSCR GGCWGCTRC)

[α] Zinc fingers are named after the four amino acids expected to contact DNA (232).
[β] H = A, C, or T; M = A or C; N = A, C, G, or T; R = A or G; W = A or T
[γ] bp = base pairs.

additional genes were identified (data not shown).

Despite using human zinc fingers that recognize only 27 of 64 possible

DNA triplets, ESPSearch identifies possible heterodimers in half of the p53 recognition sites searched, including one as short as 20 bases. A majority of the sites where a heterodimer was not identified are only 20 bases long, and all are shorter than 27 bases. The number of recognition sites for the zinc fingers, and the possible ways to arrange them into heterodimers, leads to millions of possible arrangements. ESPSearch identifies possible targets for each of the genes in Table 2.1 by determining which sequences matched at least one of those arrangements, and does so nearly instantaneously.

2.3.2 Identification of Homodimeric Artificial Transcription Factors

We next searched for sequences expected to be bound by homodimers, because homodimers are encoded in half the DNA required to encode a heterodimer and are of particular interest for applications involving RXRα. For therapeutic applications it is often desirable to deliver as little DNA as possible. Homodimers are a more challenging design problem, because the likelihood of finding a target sequence is dramatically reduced. However, as nuclear receptors only naturally contain two zinc finger domains (47), the additional complexity can be compensated for. To identify a variety of hits, the search pattern was changed to contain only two zinc fingers in each monomer, allow a greater gap between units, and identify dimers in three orientations.

ESPSearch simultaneously searched for direct, inverted, and everted repeats recognized by human zinc fingers, all of which RXRα is known to recognize under some circumstances (7, 47). Specifically, the patterns $AB_{0\text{-}15}AB$ and $AB_{0\text{-}15}ba$ were used, which correspond to homodimers containing two zinc fingers recognizing direct repeats (ABAB) and inverted (ABba) or everted repeats (abBA), with a variable gap of 0-15 bases (Figures 2.1B and 2.1C). We chose *BAX* as the gene to be regulated, and used the 1,000 base pairs before the transcription start site as the source sequence.

Table 2.2. Homodimers and recognized sequences in the *BAX* promoter matching the patterns $AB_{0\text{-}15}AB$ or $AB_{0\text{-}15}ba$.

Identified Zinc Fingers for Monomers[α]	Recognized Sequences[β]	Repeat Type	Position[γ]
DSCR;RDER2	GCGGCC N_{10} GCGGCC	Direct	-68
RDHR1;RDER2	GCGGGG N_{13} GCGGGG	Direct	-76
RDER2;QSSR1	CGCTRC N_{3} GYAGCG	Everted	-80
DSAR;DSCR	GCCRTC N_{5} GCCRTC	Direct	-197
RDHT;QAHR	GGANGG N_{5} GGANGG	Direct	-198
RDHT;QAHR	GGANGG N_{9} GGANGG	Direct	-202
QSNI;DSAR	RTCMAA N_{9} RTCMAA	Direct	-204
HSNK;QSNT	AAARAC N_{11} GTYTTT	Inverted	-376
HSNK;QSHT	HGARAC N_{3} GTYTCD	Inverted	-565
RDER1;QSSR1	GYAGHG N_{14} CDCTRC	Inverted	-612
VSTR;KSNR	GAGGCW N_{7} GAGGCW	Direct	-652
ISNR;VSTR	GCWGAW N_{2} WTCWGC	Inverted	-909

[α] Zinc fingers are named after the four amino acids expected to contact DNA (232).
[β] D= A, G, or T; H = A, C, or T; M = A or C; N = A, C, G, or T; R= A or G; W = A or T, Y = C or T.
[γ] Relative to transcription start.

The p53 recognition site occurs in the middle of this region (238). We are unaware of any property that makes homodimers particularly easy or difficult to identify in the *BAX* promoter; other genes may require more or less stringent patterns to identify a reasonable variety of targets.

Table 2.2 shows the identified sequences and corresponding zinc finger proteins after discarding hits in a likely non-unique region of AT repeats. No sequence overlaps the p53 recognition site, but several of them are within approximately 100 base pairs. ESPSearch identified numerous possible dimeric artificial transcription factors in this relatively short promoter region, of a variety of types. Dimeric activation domains often prefer a particular orientation, so the ability to chose different repeat types

allows for many different activation domains to be used in the artificial transcription factor. The ability of ESPSearch to identify numerous possible target sequences for these zinc finger constructs suggests that natural human zinc fingers may be sufficient for activating any gene, without the need for engineering specificity, provided a tool is available to analyze the DNA sequence and identify possible target sequences.

2.3.3 Estimating the Specificity of Artificial Transcription Factors

To reduce possible side effects in therapeutic applications, the artificial transcription factor should be highly specific for the desired target site. A count of the occurrences of other likely binding sites in the genome is a reasonable estimate of the specificity. To identify the most promising potential artificial transcription factor for each gene in Tables 2.1 and 2.2, we used ESPSearch to search the entire human genome for each identified sequence. A target sequence database of the binding sites for the dimers was constructed, and individual GenBank® chromosome sequences of draft 34 of the human genome were searched; ESPSearch can automatically search multiple files for the same set of target sequences. No pattern processing was performed during this step. The resulting number of sites likely to be bound by each dimeric artificial transcription factor are shown in Table 2.3. Using this information, we can distinguish between the multiple choices for each gene and choose the artificial transcription factor that is likely to be most specific.

A typical search speed with ESPSearch was approximately one million bases per second, with peak speeds of nearly three million bases per second (data not shown). Artificial transcription factor binding sites were identified in the whole genome in approximately twenty minutes.

Table 2.3. Number of occurrences of dimer recognition sequences in the human genome.

Sequence[α]			Occurrences
GCWRTCGCW		GGGRACGHG	1
RTCNGGRGA	N_5	GGAHGAMAA	16
RGAMAAGCC		NGGGCGNGG	9
MAAGCCNGG		GCGNGGGCW	4
MAAGTTRGA	N_2	MAAGCCNGG	8
GTTRGARAC	N_2	GCCNGGGCG	40
RGAMAAGCC	N_3	GCGNGGGCW	1
MAAGTTRGA	N_5	GCCNGGGCG	3
GTTRGARAC	N_5	NGGGCGNGG	5
HGAMAAGCC		NGGGCWRGA	43
HGAMAAGCC	N	GGGMAAGAG	6
GAGMAAGCC		CCNGGCWGC	5
GAGMAAGCC	N_3	GGCWGCTRC	2
GCGGCC	N_{10}	GCGGCC	270
GCGGGG	N_{13}	GCGGGG	1,219
CGCTRC	N_3	GYAGCG	46
GCCRTC	N_5	GCCRTC	441
GGANGG	N_5	GGANGG	41,438
GGANGG	N_9	GGANGG	47,304
RTCMAA	N_9	RTCMAA	12,503
AAARAC	N_{11}	GTYTTT	5,958
HGARAC	N_3	GTYTCD	91,618
GYAGHG	N_{14}	CDCTRC	375,440
GAGGCW	N_7	GAGGCW	2,781
GCWGAW	N_2	WTCWGC	4,646

[α] D = A, G, or T; H = A, C, or T; M = A or C; N = A, C, G, or T; R = A or G; W = A or T; Y = C or T.

2.3.4 Estimating the Specificity of siRNAs

A similar approach can be applied to determine the likely specificity of siRNAs. Recent work suggests that the common practice of using BLAST to determine the uniqueness of siRNA binding sites fails to identify many

possible target sites (242). BLAST identifies sequences in the source sequence homologous to the target sequence using a scoring function. The result is not necessarily the same as a search for a sequence with specific variation. For example, BLAST produces different scores for sequences depending on the location of mismatches: mismatches on the ends penalize a sequence slightly, mismatches as a group in the middle of a sequence penalize the sequence moderately, and mismatches scattered throughout the sequence penalize the sequence heavily. Depending on the exact parameters used for the BLAST search, some sequences may not be identified that have the same number of mismatches as sequences that are identified. ESPSearch evaluates a sequence strictly on how many mismatches occur, not where they occur, and so will find all sequences containing a particular number of mismatches.

We arbitrarily chose six 21-mers from the *BAX* mRNA as test sequences for siRNA specificity, and used three mismatches as the threshold for possible binding. We used both NCBI's BLAST (version 2.2.9) and ESPSearch to search for each sequence in the whole human genome and in the set of known and predicted human mRNAs. The number of matches identified by each program are shown in Table 2.4. In should be noted that in every mRNA and genome search, BLAST found hundreds to thousands of sequences, but the majority were sequences with greater than three mismatches. ESPSearch identified only sequences with three or fewer mismatches, and found many missed by BLAST.

ESPSearch took approximately fifty minutes to identify hits on the whole genome and less than ninety seconds on the mRNAs. Individual BLAST searches were typically ten-fold or more faster than ESPSearch searches, but the time required to compile the BLAST database (in the case of mRNA, run using stand-alone BLAST) or wait for the web server response

Table 2.4. Number of siRNA sequences found with BLAST and ESPSearch. Searches results are divided into exact matches and matches containing up to 3 mismatches.

	Human Genome				Human mRNA			
	BLAST[α]		ESPSearch		BLAST[α]		ESPSearch	
Sequence	**0**	**1-3**	**0**	**1-3**	**0**	**1-3**	**0**	**1-3**
GCGGCGGTGATGGACGGGTCC	1	0	1	3	6	0	6	1
ATGGGGGGGGAGGCACCCGAG	1	1	1	34	0	0	5	0
AGGATGATTGCCGCCGTGGAC	1	2	1	7	4	1	4	2
AGCAAACTGGTGCTCAAGGCC	0[β]	11	0[β]	63	4	1	4	4
TGGGTGAGACTCCTCAAGCCT	1	5	1	49	1	0	1	1
TGGTGCCCTCTCCCCATCTTC	1	16	1	177	1	0	1	5

[α] Using NCBI's default "short or nearly exact sequence" parameters: word size = 7, expect = 1000.
[β] No hits are found because the final two bases of this sequence are on a separate exon from the other bases in the sequence.

(in the case of whole genome search) meant that the total time investment was comparable for the two programs. Because finding genes that might be affected by siRNAs is critical for evaluating their effectiveness, especially for medical applications (242), ESPSearch is a superior tool for this application.

2.3.5 Identifying Possible Protein-Protein Binding Sites

We also applied ESPSearch to determine the frequency of protein motifs in the human proteome. The nuclear receptor family of transcription factors is known to interact with coactivators via an LxxLL motif present in coactivators (22). Chang and coworkers (243) used phage display libraries to determine the importance of residues flanking the core LxxLL region. Three different classes of LxxLL peptides were found, each with different activity; the three classes vary in the three amino acids N-terminal to the LxxLL core. We searched the non-redundant set of human proteins for the LxxLL motif and the three identified classes. Table 2.5 shows the resulting number of hits identified for each motif and the number of proteins

Table 2.5. Number of occurrences of LxxLL motifs and proteins containing the motifs.

Sequence[α]	Occurrences	Number of Proteins
LXXLL	54,255	28,554
SRLXXLL	212	212
HPLLXXLL	27	27
UJLXXLL	1,824	1,742

[α] J = I, L, or V; U = S or T; X = any amino acid.

containing at least one of each motif. A casual survey of the list of proteins identified for each of the three peptide motifs identified by phage display turns up several coactivators, suggesting that these motifs may play a role in natural proteins as well. This type of search was not performed in the original study, but took less than thirty seconds with ESPSearch. ESPSearch can not only rapidly identify protein motifs, but can also be used to determine if a hypothetical, designed, or library-selected motif is present in known proteins of a particular type.

2.4 Discussion

ESPSearch is able to identify essentially any target sequence within any source sequence to find the exact desired sequence(s). Specific strengths include searching any number of sequences simultaneously, identifying target sequences of any length and considerable complexity, the ability to match patterns composed of hits from individual target sequences, and the ability to easily modify the databases, wildcards, and rules to allow user-defined groups and non-standard nucleic acids or amino acids. In addition, ESPSearch functions without a network connection, requires no knowledge of regular expression design, and can have functionality added or modified if necessary.

ESPSearch may not be the best choice for specific applications where

specialized tools exist for the search. BLAST is a much superior tool for general alignments, especially when it is not practical to designate all variability (e.g., that a gap can appear anywhere). ESPSearch is a generalized tool useful for user control and many applications, but it therefore lacks some of the specialized features available using TRANSFAC® or other tools focused on a more specific application. ESPSearch may be slower than specialized tools for some searches and it does not analyze the output, although it is straightforward to feed the resulting output to a spreadsheet or another program for analysis.

We have demonstrated the identification of target sites for and design of artificial transcription factors containing human zinc fingers, estimated the genome-wide specificity of the possible artificial transcription factors and a variety of siRNAs, and identified possible sites for protein-protein interactions. The limiting temporal factor for these applications has been reduced to obtaining source sequences to search, not ESPSearch searching the sequences or identifying patterns. ESPSearch should be useful for identifying DNA target sequences when engineering transcription factors, enzymes, and even small molecules, as well as when designing small interfering RNAs. In addition, ESPSearch could be used to identify almost any nucleic acid or protein motif by using the corresponding target sequences and patterns, without the need to design or use a specialized tool. It can be used as an alternative to alignment tools such as BLAST when exact control over the sequence is required, and is fast enough to do whole genome scans for many sequences and patterns of those sequences for various analyses: ESPSearch can currently search the human genome in as little as a few minutes, and the yeast genome in seconds. The program was recently adapted by researchers at the University of Georgia for use via the web with RoseoBase (http://www.roseobase.com) to search for short

sequences in the Roseobacter database. ESPSearch could also be applied to analyzing the frequency of particular sequences in the human genome, determining where specific transcription factor binding sites occur in relation to each other, and identifying cofactor binding sites in proteomes.

CHAPTER 3
EXPRESSION AND PURIFICATION OF hRXRα

3.1 Introduction

Retinoid X receptor α, and portions thereof, have previously been expressed and purified in a variety of systems: mammalian (176, 216, 244-251), baculovirus (insect) (20, 177, 178, 247, 250, 252, 253), yeast (253), and bacterial (21, 176, 187, 188, 190-193, 199, 201, 210-214, 247-250, 254-276) cells, as well as using *in vitro* expression (277). Full-length RXRα has not been expressed to high levels in any of these systems, as is common to many nuclear receptors (278). All work with biophysical characterization of RXRα(ABCDE) (see Section 1.2 for nomenclature) has been done using unpurified protein in cellular extract, generally from mammalian cells (20, 176-178, 216, 244-247, 249, 250, 252, 261) or *in vitro* expression (277). The majority of work has been done on specific domains: RXRα(C) (212-214, 255); RXRα(CD) (210, 211); RXRα(CDE) (21, 188, 247, 248, 250, 257-259, 261-264, 270, 271); RXRα(DE) (187, 249, 256, 272, 273); and RXRα(E) (190-192, 199, 201, 249-251, 254, 260, 265, 266, 268-270, 274-276). There are also a few reports of isolation of domains from USP (193, 205-209, 267, 279, 280), RXRβ (185, 202, 203, 251, 281-283), and RXRγ (186).

The approaches for RXRα expression and purification taken to date suffer from three problems. First, the majority of protocols call for two or more chromatographic steps (21, 188, 190-193, 199, 201, 247, 250, 257-259, 261, 262, 264-269, 271, 272), which are very time-intensive. The first step is generally metal-based affinity chromatography (284) using a His_6 tag, although a few investigators have used glutathione-s-transferase (GST) tags (249, 254, 270, 276). Second, the yields are relatively low. When reported, they have been in the range of 5-15 mg L^{-1} of culture (247, 248, 256, 260);

because those protocols reporting yields are often the basis for the others, it is unlikely any have significantly improved upon these yields. The exception is expression done in extremely rich terrific broth media, where up to 45 mg L^{-1} has been reported (270). Third, the expressions in bacteria have frequently led to inconsistent and contradictory results. For example, some authors report explicitly that they observe batch-to-batch variation in protein behavior (247, 249). In addition, although it is generally not commented on in the context of expression variability, there are a wide range of oligomeric states and behaviors reported for RXRα. Specifically, some groups isolate what they identify as a pure dimeric species (192, 201, 248, 256, 265, 266, 268, 269, 275), while others observe dynamic equilibrium between monomer and dimer (244), dimer and tetramer (188, 261, 262, 271, 272), monomer and tetramer (20, 250), and all three states (21, 257, 258). As a careful review of the literature shows, even individual research groups are not consistent with the observations they report, suggesting batch-to-batch variations. A large number of reports simply ignore this issue altogether, although it is not clear that this approach is justified.

We desired an expression and purification scheme that would address the problems described above. The most important goal was to ensure that batch-to-batch consistency was extremely high. As we also wished to apply this system to a large number of RXRα variants, a single chromatographic step generating at least 95% purity was desirable. Finally, to minimize the time and effort required during expression, we sought a significant increase in the yield of purified protein. To meet these requirements, both the expression and purification procedures for hRXRα($D_{\beta}E$) were extensively optimized, and additional work was performed using other hRXRα domains.

3.2 Materials and Methods

3.2.1 Materials

The oligonucleotides used for cloning were as follows: F1, 5'-CTA GCT AGC AAC ACC AAA CAT TTC CTG CCG-3'; F2, 5'-CAT GCC ATG GGC AAG GAC CGG AAC GAG-3'; F3, 5'-CTA GCT AGC TGC GCC ATC TGC GGG-3'; F4, 5'-CTA GCT AGC AAG CTA CTG TCT TCT ATC GAA CAA GC-3'; R1, 5'-CCG CTC GAG AGT CAT TTG GTG CGG CGC-3'; R2, 5'-CCG CTC GAG ATC ACT ACG CTG CCG CTC CTC CT-3'; R3, 5'-CCC AAG CTT AGT CAT TTG GTG CGG CGC-3'.

The plasmids containing the pregnane X receptor gene (pRSETAPXRW299M) and coactivator fragment (pACYC184_SRC1(623-710)) were generous gifts from Dr. Matt Redinbo.

SYPRO® Red was purchased from Molecular Probes (Eugene, OR). Pfu, QuikChange® Mutagenesis kits, BL21-CodonPlus®-RIL cells, and BL21-CodonPlus®-RP cells were obtained from Stratagene (La Jolla, CA). All other enzymes were obtained from New England Biolabs (Ipswich, MA). Gel DNA Recovery Kits and Z-competent™ Transformation Kits were obtained from Zymo Research (Orange, CA). pET28a was purchased from Novagen (San Diego, CA). Miniprep kits were obtained from Qiagen (Valencia, CA). BL21(DE3)pLysS cells were purchased from Invitrogen (Carlsbad, CA). The PRO™ Tet 6xHN Bacterial Expression System and Talon® Co^{2+} resin were purchased from Clontech (Mountain View, CA). Gel electrophoresis was performed using the Bio-Rad (Hercules, CA) Mini-Protean® III system. Trypsin and ProteoPrep™ reduction and alkylation kits for mass spectrometry were obtained from Sigma-Aldrich (St. Louis, MO).

3.2.2 Cloning of hRXRα Domains

hRXRα($D_{\beta}E$) (amino acids 212-462) was amplified from a vector

containing hRXRα using oligonucleotides F2 and R1 and Pfu to create the gene insert. The insert was digested with NcoI and XhoI and gel cleaned. pET28a was similarly digested, treated with calf intestinal alkaline phosphatase, ligated to the insert using Quick Ligase, transformed into Z-competent™ XL1-Blue *E. coli,* plated on Luria-Bertani (LB) media containing 30 μg mL^{-1} kanamycin, and grown overnight at 37 °C. The resulting pET28-hRXRα($D_{\beta}E$) plasmid was isolated via miniprep and confirmed by sequencing.

Alternative constructs were cloned similarly. hRXRα(ABCDE) was amplified from the hRXRα vector with oligonucleotides F1 and R1 and digested with restriction enzymes NheI and XhoI, and ligated into similarly digested pET28a. hRXRα($ABCD_{\alpha}$) was cloned identically, except R2 was substituted for R1, and then QuikChange® mutagenesis was used to correctly insert two stop codons following residue 212, using primers 5'-GGT GGT GGT GGT GCT CGA GCT ATC ACG CTG CCG CTC CTC C-3' and 5'-GGA GGA GCG GCA GCG TGA TAG CTC GAG CAC CAC CAC CAC C-3'. hRXRα(CDE) was first created using Quick-Change mutagenesis to delete the A/B domain of hRXRα using primers 5'-CCC AAG CTT CTA GGT ACC ATG TGC GCC ATC TGC GG-3' and 5'-CCG CAG ATG GCG CAC ATG GTA CCT AGA AGC TTG GG-3'. hRXRα(CDE) was then amplified with primers F3 and R1, digested with NheI and XhoI and ligated into similarly digested pET28a. GBDhRXRα(DE) was amplified from a plasmid containing GBDhRXRα(CDE) with primers F4 and R3, digested with NheI and HindIII, and cloned into pET28a. QuikChange® Mutagenesis was then used to remove the RXRα DBD from GBDhRXRα(CDE) to create the final GBDhRXRα(DE) using primers 5'-GTT GAC TGT ATC GCC GGA ATT CAT GAA GCG GGA AGC CGT G-3' and 5'-CAC GGC TTC CCG CTT CAT GAA TTC CGG CGA TAC AGT CAA C-3'.

3.2.3 Expression and Purification of hRXRα($D_\beta E$)

Detailed here is the final, optimized protocol for hRXRα($D_\beta E$); for details of alternative approaches and other constructs, see Section 3.3.2.

For expression, pET28-hRXRα($D_\beta E$) was transformed into Z-competent™ BL21(DE3) and plated on fresh LB media containing 30 μg mL^{-1} kanamycin and grown overnight at 37 °C. 5 mL of LB containing 30 μg mL^{-1} kanamycin was inoculated with a single colony and grown in 5 mL of overnight at 37 °C with shaking. 250 mL of LB containing 30 μg mL^{-1} kanamycin was inoculated with 2.5 mL of the overnight culture and grown at room temperature (20-25 °C) with vigorous shaking in a 1 L Erlenmeyer flask until the OD_{600} was 0.4-0.6 (approximately three hours). Isopropyl-β-D-thiogalactopyranoside (IPTG) was then added to a final concentration of 0.5 mM, and the culture grown under the same conditions for a further 20-24 hours. Cells were harvested by centrifuging for 20 minutes at 3,000 xg at 4 °C. The media was removed and the cell pellet was immediately used for purification.

The cell pellet was resuspended in 5 mL of 4 °C Lysis Buffer (300 mM NaCl, 50 mM NaP_i, 10 mM imidazole, pH 7.0) in a 15 mL polypropylene tube. Lysozyme was added to a final concentration of 250 μg mL^{-1}. The mixture was kept on ice for 30 minutes, then sonicated using a Fisher Scientific Model 60 Sonic Dismembrator (Pittsburgh, PA). Sonication was performed on ice for 3x10 seconds with 30 second pauses between pulses, at an average power of 12 watts. The lysed cells were then centrifuged for 20 minutes at 27,000 xg at 4 °C. The insoluble protein content was estimated by resuspending the cell pellet in Lysis Buffer containing 1% SDS, mixing vigorously for 60 seconds, allowing the tube to sit at room temperature for 20 minutes, then mixing again and centrifuging at 27,000 xg for 20 minutes. The supernatant then contained all the remaining hRXRα($D_\beta E$) present in

the pellet, and more aggressive lysis procedures did not increase the hRXRα($D_\beta E$) content despite an increase in the amount of total protein.

hRXRα($D_\beta E$) was purified by Co^{2+} (Talon®) affinity chromatography, with all steps performed at 4 °C; the following details are calibrated for 0.5 mL (bed volume) resin, but all volumes can be doubled for 1.0 mL resin. 0.5 mL of Co^{2+} resin was transfered to a 15 mL polypropylene tube and prepared by washing with 2x5 mL of Lysis Buffer, centrifuging for 2 minutes at 700 xg after each wash and removing the buffer. The cell lysis supernatant was decanted onto the resin and gently mixed on an orbital rotator for 45 minutes. The resin was pelleted by centrifugation at 700 xg for 5 minutes and the supernatant removed. The pellet was then washed twice with 5 mL of Wash Buffer (300 mM NaCl, 50 mM NaP_i, pH 7.0) for 20 minutes on an orbital rotator, centrifuging at 700 xg for 5 minutes after each wash and removing the supernatant. The cell pellet was transfered to a column housing by adding 1 mL of Wash Buffer, gently mixing, and pipetting the slurry to the housing. The resin was allowed to settle for 5 minutes, packed by draining most of the Wash Buffer, and washed with a further 5 mL of Wash Buffer. The protein was then eluted using 3.5 mL of Elution Buffer (300 mL NaCl, 50 mM NaP_i, 150 mM imidazole, pH 7.0) in a single fraction. The elution fraction was then dialyzed into at least 250 mL of Storage Buffer (500 mM NaCl, 10 mM NaP_i, 35% glycerol, 1 mM tris(2-carboxyethyl) phosphine, pH 7.0) using 10,000 molecular mass cutoff dialysis tubing, changing the buffer once after at least four hours and then dialyzing overnight. The recovered protein, generally concentrated approximately three-fold due to the high glycerol content in the dialysis buffer, was stored at -20 °C.

The Talon® resin was rinsed with 5 mL of 50 mM 2-(N-morpholino)ethanesulfonic acid (pH 5.0) and 5 mL of dH_2O, then mixed with

0.5 mL of 0.1% sodium azide in 20% ethanol and stored at 4 °C. Aliquots of resin were used at most twice for a single protein before stripping the metal and recharging for additional use, following the instructions in the Talon® manual.

SDS-PAGE of the purified protein was done using 0.05% SDS in tris-glycine buffer through a 12% (29:1 acrylamide:bisacrylamide) polyacrylamide gel with a 4% polyacrylamide stacking gel. SYPRO® Red was used to stain gels.

3.3 Results

3.3.1 Initial Characterization of hRXRα(DβE)

The final concentration of the protein in the storage buffer was determined by absorption at 280 nm (285) using a molar absorptivity of 16960 cm^{-1} M^{-1}. Following the procedure detailed in Section 3.2.3, the final hRXRα(DβE) concentrations from several purifications were 240-300 µM (7.0-8.7 mg mL^{-1} in 1.3 mL), indicating a yield of 36-45 mg L^{-1} of culture. Purity of the protein was determined by SDS-PAGE, using multiple concentrations of protein, including a massively overloaded lane. A typical gel is shown in Figure 3.1. Identity of the protein as hRXRα(DβE) was confirmed by digestion with trypsin and mass spectrometry. The resulting peaks were matched to those expected from a theoretical digest for hRXRα(DβE). In addition, a database search was performed against the most intense signals, and the only statistically significant match was hRXRα.

As detailed in Section 4.3, several different batches of hRXRα(DβE) and variants had identical binding, thermal, and oligomeric behavior.

3.3.2 Factors Influencing Yield and Purity

Many factors are important in maximizing the purity and yield of hRXRα(DβE) throughout the procedure, while others have little or no

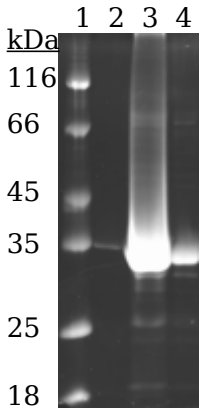

Figure 3.1. Purity of hRXRα($D_{\beta}E$). Lane 1 is molecular weight markers. Lane 2 is 50 ng of protein, to estimate intensities of weak bands. Lane 3 is 25 µg of protein, to show contaminants as low as 0.1% the total concentration. Lane 4 is 2.5 µg of protein, showing contaminants at the 1 % level.

influence on the outcome. Here we outline the alternative approaches we explored before settling on the final procedure in Section 3.2.3.

Expression using strains of *E. coli* other than BL21(DE3) or in alternative expression systems significantly reduced expression levels. The BL21(DE3)pLysS strain is designed to reduce pre-induction expression and aid in lysis, but the pLysS strain had a significantly decreased yield compared to BL21(DE3). No pre-induction expression of hRXRα($D_{\beta}E$) was detectable in any strain at temperatures below 25 °C. Results using the BL21-CodonPlus® strains, which are designed to increase expression when genes (such as RXRα) contain codons that are infrequently used by the expression organism, were similar to those for BL21(DE3)pLysS. The PRO™ Tet 6xHN expression system failed to yield detectable protein levels.

Media and temperature also had dramatic impact on the expression levels. Expression in M9ZB minimal media, which aided in expression of full-length vitamin D receptor (286), led to very large reductions in yield. Richer media than LB, such as terrific broth, did not increase yield of soluble protein, although total protein content did increase. Regardless of media

type, room temperature was found to be optimal for expression levels, as temperatures below 20 °C reduced bacteria growth significantly, while significant background of insoluble protein was observed at temperatures above 25 °C prior to induction, which seems to have triggered expression of predominantly insoluble protein following induction.

We tried two approaches to improve the protein folding. Adding 6% ethanol upon induction is expected to trigger stress and production of chaperone proteins (287). Although the insoluble hRXRα($D_{\beta}E$) was reduced to undetectable levels when ethanol was added, the overall yield was also reduced by approximately 70%. Other nuclear receptors, such as the pregnane X receptor, benefit from the presence of a coactivator protein fragment containing an LxxLL motif (288). However, while we were able to reproduce the effect of the motif on the pregnane X receptor (i.e., the protein only expressed in the presence of the LxxLL motif), the coactivator fragment had the opposite effect for hRXRα($D_{\beta}E$): it completely eliminated expression.

Several variables were of only minor importance. The IPTG concentration used for induction needed to be approximately 0.5 mM for maximum expression, but higher IPTG concentrations had no additional effect. The age of the colony used for the expression was not important, provided the plate used was not so old that the cells were mostly dead (i.e., greater than one month). The OD_{600} at induction was not a major factor, and the same final yield was observed with an induction at an OD_{600} anywhere in 0.15 to 1.0 range.

Several factors were tested during cell lysis. The lysozyme concentration was the least important, as it helped lysis only slightly at 250 μg mL^{-1}, and higher concentrations were no more effective. Sonication provided most of the lysis power, but we used less sonication than generally

called for, as in the Talon® manual and in the RXRα purification protocols with explicit details (188, 249). The 3x10 second protocol consistently gave > 75% lysis, and usually essentially complete lysis, while not damaging the protein. Sonication was optimal when performed in relatively small volumes (< 7 mL) and in as vertical a liquid column as possible (i.e., a narrow 15 mL tube was superior to a wider 50 mL tube). Performing the lysis at 4 °C throughout the process rather than room temperature seemed to prevent conversion of soluble protein to insoluble. Finally, freezing the cell pellet at -80 °C for at least 12 hours was very effective in aiding lysis, but the half-life of soluble protein at -80 °C was less than 24 hours due to conversion to insoluble aggregates. We therefore avoided freezing due to the large yield penalty and the possibility of more subtle effects on structure.

Purification was tolerant to variations, with most variables being of only minor or insignificant importance. The specific buffer (tris or phosphate), pH (7-8), salt concentration (100-500 mM), glycerol concentration (0-10%), and presence of reducing agents or protease inhibitors all had negligible effects on the final yield. Performing the purifications at room temperature instead of 4 °C led to a small drop in both yield and purity. Use of polystyrene instead of polypropylene tubes also caused a small drop in yield, possibly due to adsorption of hRXRα($D_{\beta}E$) to the polystyrene.

A few variables were fine-tuned to balance yield and purity. A resin bed of approximately 0.5 mL per 250 mL culture gave maximum purity while retaining the large majority of the protein. Although pooling two 250 mL cultures together and doubling the resin gave comparable results, but pooling more than two cultures into a single purification decreased purity significantly, possibly due to less efficient mixing. In addition, the use of 10 mM imidazole in the Lysis Buffer and a longer equilibration time on the

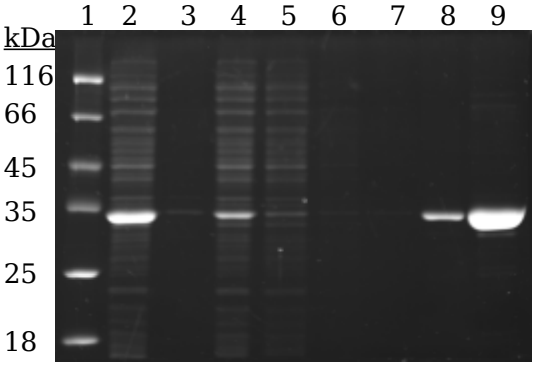

Figure 3.2. Monitoring the purification of hRXRα($D_{\beta}E$). Lane 1, molecular weight markers. Lane 2, soluble fraction. Lane 3, insoluble fraction. Lane 4, soluble fraction after equilibration with resin. Lanes 5 and 6, sequential washes. Lane 7, pre-elution flow-through after loading resin into the column housing. Lane 8, resin flush with low pH after elution. Lane 9, elution fraction. Note that lanes 2-4 are diluted five-fold with respect to the other lanes.

resin dramatically improved purity while not decreasing yield, most likely by providing appropriate conditions for hRXRα($D_{\beta}E$) to occupy the maximum number of binding sites. The use of imidazole at later steps prior to elution was not helpful in increasing purity further and led to significant loss of protein during washes.

Figure 3.2 shows a typical gel monitoring the purification procedure, illustrating that the overwhelming majority of protein is in the soluble fraction and that most becomes bound to the resin. Based on the weak band in Lane 2, and that all the hRXRα($D_{\beta}E$) in the insoluble fraction was recovered with a relatively mild procedure (see Section 3.2.3), we suspect most of the "insoluble" protein was small amounts left behind after decanting the lysis supernatant and in a small number of unlysed cells. Figure 3.2 also shows that some protein remains bound following elution (Lane 8), but this fraction may be of a different conformation than the eluted fraction, as it only elutes in the presence of low pH and not imidazole. The very weak additional bands present in the elution fraction could not be

eliminated by any modification to the protocol. Additional gel filtration or ion exchange chromatography following the affinity chromatography did not eliminate these bands or significantly improve purity.

3.3.3 Expression of Alternative Constructs

In addition to the hRXRα($D_\beta E$) construct, nine hRXRα($D_\beta E$) variants containing up to five mutations were purified using the same procedure. Expression of four different combinations of domains was also explored: hRXRα(ABCDE), hRXRα($ABCD_\alpha$), GBDhRXRα($D_\beta E$), and hRXRα(CDE). In all cases expression was initially attempted using the protocol developed for hRXRα($D_\beta E$), then modified as described below for each construct.

The hRXRα($D_\beta E$) variants generally behaved similarly to the wild-type protein. With one exception, yields were at least as good as for wild-type (35-80 mg L^{-1}), and purities were over 95%. The sole exception gave yields of 10-15 mg L^{-1} and purities of approximately 90%, and has significantly different self-association properties from the other receptors (see Chapter 5).

hRXRα(ABCDE) had very low levels of expression under all conditions, and induction generally led to a rapid halt in bacterial growth. Although very low levels of protein were detectable in 1-2 hours, additional time did not increase the quantity of protein. Use of minimal media, addition of 6% ethanol, addition of the coactivator fragment, or the presence of zinc chloride generally reduced expression to undetectable levels, and in all cases cellular growth was halted. At 37 °C, the cells did not seem affected by induction (i.e., they continued to grow rapidly), but expression of hRXRα(ABCDE) was undetectable. hRXRα($ABCD_\alpha$) showed essentially the same behavior as hRXRα(ABCDE) under all conditions, except that it tended to express at slightly higher levels.

hRXRα(CDE) expressed well using the hRXRα($D_\beta E$) protocol, and

could be purified easily using tris instead of sodium phosphate (to avoid zinc phosphate precipitation). Yields were as high as 80 mg L^{-1}. However, addition of equimolar zinc following purification led to rapid precipitation of the protein, which was reversed upon addition of ethylenediamine tetraacetic acid. This behavior was observed regardless of whether or not hRXRα(CDE) was expressed in the presence of zinc chloride. Purification using Talon® resin with Zn^{2+} substituted for Co^{2+} also induced precipitation.

GBDhRXRα(DE), which contains the yeast Gal4 DNA binding domain (GBD) fused to the hinge and ligand binding domain of hRXRα, is used by the Doyle laboratory in genetic selection for hRXRα variants (218-221). It was successfully expressed to levels similar to hRXRα(CDE), and could easily be purified without precipitation. However, BL21(DE3) apparently contains a protease that cleaves in the middle of the GBD, as the purified product was a mixture of the expected product (minor) and a truncated product (major) approximately midway in size between GBDhRXRα(DE) and hRXRα($D_{\beta}E$). Use of protease inhibitors during purification did not influence the final yield, suggesting that the protease is present during expression. Expression at 37 °C produced exclusively truncated protein. Expression at 15 °C did not change the fraction of truncated protein relative to our standard expression protocol at 22 °C.

3.4 Discussion

Compared to previous purification protocols for RXRα containing the LBD (21, 187, 188, 190-193, 199, 201, 247-250, 254, 256-258, 260-276), our approach has several advantages. First, the C-terminal His_6 tag ensures that no incompletely translated products are co-purified with the intended protein target. Although the C-terminal His_6 tag might be expected to influence the structure of function of the protein, it is not clear that an N-terminal tag attached to the hinge is any "safer," due to recent reports

suggesting important roles for the hinge region in ligand binding and activation (48-50). Both the N- and C-termini are disordered in RXRα crystal structures (189-200), so neither terminus is an obvious choice. Moreover, by keeping the tag as short as possible (LEHHHHHH), we minimize the risk of undesired effects; even designing a longer tag and cleaving using a protease generally leaves several amino acids. For example, in at least one prior RXRα report, twenty extra amino acids were still attached to the N-terminus following tag cleavage (249). Thrombin, the most common enzyme used to cleave tags, leaves a minimum of four amino acids, and some pET plasmids also have a T7 tag, which leaves twelve amino acids after cleavage.

Second, relative to the more commonly used ampicillin marker plasmids (21, 187, 190-192, 199, 201, 247-250, 254, 256-276), the pET28a plasmid has the advantage of lower background expression due to the reversed orientation of the gene, preventing simple read-through from transcription of the antibiotic resistant gene; to date only two other reports of pET28-based RXRα expression exist (188, 193). The use of kanamycin instead of ampicillin may also offer an advantage, as kanamycin works by inhibiting the ribosome and thus ensures that all cells are constantly under selection and reliably producing protein, whereas ampicillin is deactivated extracellularly and only ensures that enough cells retain the expression plasmid to convey resistance to all cells.

Our protocol also simplifies the purification procedure. Nearly all prior purification protocols required at least two chromatographic steps (21, 188, 190-193, 199, 201, 247, 250, 257-259, 261, 262, 264-269, 271, 272). Unlike many previous investigators, by optimizing the affinity chromatography, we obtained very high purity from a single step, and that an additional gel filtration step simply produced a loss in yield with no further purification. Most of our improvement in yield, compared to prior

protocols, is probably due to the elimination of subsequent chromatographic steps. Although we see a mixture of oligomers during gel filtration chromatography, the peaks were overlapping and not cleanly separable; repeated chromatography did not change the distribution, and which suggest an equilibrium of two states rather than distinct conformations. The clear equilibrium could be due to the C-terminal His_6 tag, which eliminates nearly-complete protein sequences that may adopt a slightly different fold from full-length protein, avoidance of freezing, and/or minimization of sonication. Finally, use of SYPRO® Red as a stain to check for purity ensured much higher sensitivity than the commonly used Coomassie stains (289), and so provided confident purity estimates.

The yield of hRXRα($D_\beta E$) from the affinity chromatography is nearly double the expected yield (10 mg mL^{-1} of resin, according to the Talon® manual). This result is not entirely unexpected, as RXRα forms dimers and tetramers (42) (see also Chapters 4 and 5). A recent report demonstrated that oligomeric proteins may bind to affinity resins using only a single His_6 tag, and so "overload" the resin (290). Moreover, under no circumstances did was yield of more than 40 mg mL^{-1} of resin (four times the expected loading) obtained, which is in line with the expectation that hRXRα($D_\beta E$) is adopting dimeric and tetrameric forms, but not higher order structures. Moreover, as detailed in Chapter 5, the single hRXRα($D_\beta E$) variant we expressed that shows reduced self-association also had dramatically reduced yield and purity.

One alternative purification strategy not pursued was that of a denaturing protocol, including possible recover from inclusion bodies. This method has been used for hRXRα(C) (212, 213), but not reported for other domains of RXRα. We felt it safer to retain a native structure throughout rather than risk improper refolding. A recent report has also demonstrated

that proteins recovered from inclusion bodies are of inferior quality (291).

We have only performed initial trials on hRXRα(ABCDE), hRXRα($ABCD_{\alpha}$), hRXRα(CDE), or GBDhRXRα(DE), and have not pursued them further. The results from these trials suggest that the A/B domain of hRXRα is likely fatal to *E. coli*. This could be due to some unpredictable protein-protein interaction, the fact that A/B is intrinsically disordered (5) and so may form a toxic aggregate, or that when the A/B domain is coupled to the DBD, it leads to inhibition of expression of an essential gene due to enhanced DNA interactions (43-46). Alternative expression systems are probably required to obtain significant quantities of hRXRα(ABCDE) and hRXRα($ABCD_{\alpha}$), although the yield may still be low. The toxicity of the A/B domain, but not the DBD, is supported by our successful expression of hRXRα(CDE), as well as similar constructs from other groups (21, 188, 247, 248, 250, 257-259, 261-264, 270, 271). In addition, there are reports of successful expression of the hRXRα DBD alone (210-214, 255), although in many cases they required the C195A mutation to prevent aggregation (cysteine 195 is not involved in DNA binding). hRXRα(CDE) is problematic because zinc is required for proper folding of nuclear receptor DBDs (212, 213, 292), but the zinc-induced precipitation suggests that some portion of the protein is improperly folded. The protease cleavage problem of GBDhRXRα(DE) could possibly be worked around by using an N-terminal His_6 tag instead of a C-terminal tag or both concurrently, but increases the risk of obtaining partially translated protein after affinity chromatography.

In summary, we have designed a single-step expression and purification system that produces hRXRα($D_{\beta}E$) of consistent quality and high yield.

CHAPTER 4
CHARACTERIZATION OF WILD-TYPE hRXRα(DᵦE)

4.1 Introduction

Although it is generally accepted that RXRα is functionally active as a dimer (7, 8, 42), and may also be active as a tetramer (262, 271), the oligomeric states that RXRα adopts are not entirely clear-cut. As mentioned in Section 3.1, the reported oligomeric state(s) of RXRα following expression and purification are quite variable. Indeed, multiple reports from single research groups are not always consistent, which demonstrates the complexity of the issue. Table 4.1 summarizes published research where some analytical measure of the oligomericity was reported and the LBD was part of the construct studied. In short, claims of a monomeric (20, 21, 244, 249-251, 256-258, 263, 294), dimeric (21, 188, 192, 201, 244, 248-251, 256-258, 262, 265, 266, 268, 269, 271, 272, 275, 293, 294), and tetrameric (20, 21, 188, 192, 201, 250, 257, 258, 262, 263, 265, 266, 271, 272) state have been made regardless of the domains investigated, with some reports noting more than one state. Some investigators observe that multiple states are separable (192, 201, 256), suggesting that one of the states is an aggregate or "defective" form; others report an equilibrium between two or three states (20, 21, 188, 244, 249-251, 257, 258, 261-263, 265, 271, 272, 294). When bound to 9cRA, investigators have reported both solely monomers (249, 250, 263) and solely dimers (21, 192, 201, 244, 248, 251, 261, 265, 266, 269, 271, 272, 293, 294), as well as an unchanged monomer-tetramer equilibrium (20) and a shift away from tetramer to a monomer-dimer equilibrium (258). The published crystal structures show similar diversity. hRXRα(DE) and hRXRα(E) (see Section 1.2 for nomenclature) have been crystallized as a monomer (189) and tetramer (191) in the absence of ligand.

Table 4.1. Published oligomerization states of RXRα.

					Oligomerization[α]		
Authors	**Year**	**Ref.[β]**	**Sp.[γ]**	**Domains**	**No Ligand**	**With 9cRA**	**K_d (nM)**
Zhang et. al.	1992	(244)	h	ABCDE	M, D	D	
Chen & Privalsky	1995	(20)	h	ABCDE	M, T	M, T	
Sanchez-Andres et. al.	2005	(294)	h	ABCDE	M, D	D	
Chen et. al.	2005	(293)	h	ABCDE	D	D	
Kersten et. al.	1997	(261)	m	ABCDE	D, T	D	
Chen et. al.	1998	(250)	m	ABCDE	M, T	M	
Fischer et. al.	2003	(188)	h	CDE	D, T	-	~100,000's
Leid	1994	(248)	m	CDE	D	D	
Kersten et. al.	1995	(21)	m	CDE	M, D, T	D	155 & 4.4
Kersten et. al.	1995	(257)	m	CDE	M, D, T	-	
Kersten et. al.	1995	(258)	m	CDE	M, D, T	M, D	130 & 2.8 63 (9cRA)
Kersten et. al.	1997	(261)	m	CDE	D, T	D	
Kersten et. al.	1998	(262)	m	CDE	T	D	
Chen et. al.	1998	(250)	m	CDE	M, T	M	
Schimerlik et. al.	1999	(263)	m	CDE	M, T	M	
Yasmin	2004	(271)	m	CDE	D, T	D	
Cheng et. al.	1994	(249)	h	DE	M, D	M	
Bourguet et. al.	1995	(256)	h	E	M, D[δ]	-	
Egea & Moras	2001	(266)	h	E	T	D	
Egea et. al.	2001	(265)	h	E	D, T	D	
Egea et. al.	2002	(192)	h	E	D, T[δ]	D	
Chen et. al.	2003	(251)	h	E	M, D	D	
Harder et. al.	2004	(269)	h	E	M, D	-	0.38 ± 0.03
Harder et. al.	2004	(269)	h	E	D	D	
Yan et. al.	2004	(268)	h	E	D	-	
Lengqvist et. al.	2005	(272)	h	E	D, T	D	
Lu et. al.	2006	(201)	h	E	D, T[δ]	D	
Yan et. al.	2006	(275)	h	E	D	-	
Chen et. al.	1998	(250)	m	E	M, D	M	

[α] M = monomer, D = Dimer, T = Tetramer, - = Not reported.
[β] Ref. = Reference.
[γ] Sp. = Species, h = Human, m = Mouse.
[δ] Non-equilibrium mixture.

Table 4.2. Published binding constants for 9cRA and RXRα.

Authors	Year	Ref.[α]	Sp.[β]	Domains	K_d (nM)	Technique
Levin	1992	(176)	h	ABCDE	9.5	Radioligand
Heyman et. al.	1992	(177)	h	ABCDE	11.7	Radioligand
Allegretto et. al.	1993	(253)	h	ABCDE	1.62 ± 0.34	Radioligand
Allegretto et. al.	1993	(253)	h	ABCDE	1.44 ± 0.40	Radioligand
Kitareewan et. al.	1996	(178)	h	ABCDE	10.2 ± 1.5	Radioligand
Lala et. al.	1996	(252)	h	ABCDE	6.7	Radioligand
Peet et. al.	1998	(216)	h	ABCDE	13 ± 3	Radioligand
Allenby et. al.	1993	(245)	m	ABCDE	15.7	Radioligand
Chen et. al.	1994	(247)	m	CDE	14 ± 6	Fluorescence
Kersten et. al.	1995	(258)	m	CDE	5 - 20	Fluorescence
Kersten et. al.	1996	(259)	m	CDE	20 ± 3	Fluorescence
Kersten et. al.	1998	(262)	m	CDE	14 ± 0.6	Fluorescence
Schimerlik et. al.	1999	(263)	m	CDE	7.6 ± 3.8	Fluorescence
Budhu & Noy	2000	(264)	m	CDE	41 ± 12	Fluorescence
Budhu & Noy	2000	(264)	m	CDE	31 ± 10[γ]	Fluorescence
Cheng et. al.	1994	(249)	h	DE	3 ± 0.5	Fluorescence
Cheng et. al.	1994	(249)	h	DE	5.6	Radioligand
Cheng et. al.	1994	(249)	h	DE	1.8[δ]	Radioligand
Cheng et. al.	1994	(249)	h	E	10 - 200	Fluorescence
Bourguet et. al.	1995	(256)	h	E	13.8 ± 1.2	Fluorescence
Li et. al.	1997	(260)	h	E	17	Radioligand
Vivat et. al.	1997	(254)	m	E	15 ± 3[δ]	Radioligand

[α] Ref. = Reference.
[β] Sp. = Species, h = Human, m = Mouse.
[γ] AF-2 region deleted.
[δ] N-terminally GST-tagged.

When bound to various ligands, hRXRα(E) has been crystallized as a monomer (190) and dimer (192). In addition, hRXRα(E) has been observed as a dimer regardless of ligand binding by NMR (201).

The variation in reported affinities for 9cRA is also quite broad, ranging from a minimum K_d of 1.44 nM (253) to 200 nM (249). Table 4.2 summarizes the previously reported K_d values for RXRα and 9cRA. An open

question is how the binding of coactivator proteins affects ligand affinity; the question has only been examined in reverse, where the effect of ligand binding on recruitment of coactivator peptides has been reported (276, 283, 295).

To address these issues, and to ensure a reliable reference point for comparison when investigating hRXRα variants, we undertook a thorough investigation of the oligomeric and ligand-binding properties of hRXRα($D_{\beta}E$). Investigating many properties at once, using a single hRXRα construct and consistent assay conditions, allows for a greater opportunity to draw correlations between observed results. Finally, to our knowledge there has been only one report examining the folding thermodynamics of hRXRα (269), so we also investigated the (un)folding of hRXRα($D_{\beta}E$).

4.2 Materials and Methods

4.2.1 Materials

hRXRα($D_{\beta}E$) was expressed, purified, and stored as previously described (Section 3.2.3). 9-*cis* retinoic acid was purchased from Biomol (Plymouth Meeting, PA) at 99% purity, which was confirmed by UV-vis absorbance spectroscopy and high performance liquid chromatography. LG335 (172) was previously synthesized in the Doyle laboratory (115). All-*trans* retinoic acid (atRA) and γ-oxo-1-pyrenebutyric acid were purchased from Sigma-Aldrich (St. Louis, MO) at 98% purity. 9cRA, atRA, and LG335 were used as concentrated stock solutions in ethanol, and concentrations determined by UV-vis spectroscopy. We used previous reported molar absorptivities for 9cRA (ε_{344} = 36,700 cm^{-1} M^{-1} (296, 297)) and atRA (ε_{350} = 45,300 cm^{-1} M^{-1} (296)), and determined the value for LG335 (ε_{255} = 22,500 cm^{-1} M^{-1}). OPBA was used as concentrated stock solutions in 1:9 dimethyl sulfoxide:ethanol, and the molar absorptivity determined in the same solvent

(ε_{354} = 18,400 cm^{-1} M^{-1}). Retinoic acid stocks were protected from light during storage and used under reduced lighting, and all ligands were stored in the dark at -20 °C when not in use. The LxxLL peptide was synthesized by the Parker H. Petit Institute for Bioengineering & Bioscience Core Facilities (Atlanta, GA). The nonsense peptide was a generous gift from Dr. Jing Li and Dr. Christoph Fahrni.

4.2.2 Ligand Binding

Fluorescence assays to determine receptor-ligand binding constants were performed using a modification of previously described methods (247, 249). Measurements were made on a Shimadzu (Hong Kong) RF-5301PC using 5 nM slits, 1 nm resolution, 1 second integration time ("slow" speed), excitation at 280 nm, and emission monitored from 300-360 nm. All assays used 50 nM hRXRα($D_{\beta}E$) in 100 mM NaCl, 10 mM NaP_i (pH 7.0), and 1 mM dithiothreitol. Each measurement was obtained by taking a baseline scan of a clean 1 cm cuvette containing buffer and ligand(s), then adding hRXRα($D_{\beta}E$) from a 10 µM stock in the same buffer to a final volume of 1 mL and a maximum of 0.2% organic solvent. After mixing and waiting 60 seconds, a single scan was taken. The cuvette was then cleaned and used for the next measurement. All measurements were taken at ambient temperature (21-23 °C), and a single 10 µM hRXRα($D_{\beta}E$) stock was used for a maximum of 1 hour or 10 measurements. For each new 10 µM hRXRα($D_{\beta}E$) stock, a reference emission scan taken in the absence of ligand was obtained. In general, measurement sets were designed to provide "interlocking" values; for example, one set of measurements included 4, 20, and 50 nM, a second set included 8, 30, and 60 nM, and a third set 12, 40, and 70 nM. This approach allows for detection of tryptophan quenching due to causes other than ligand binding.

To correct for variation in the raw fluorescence intensities between

individual sets (due to small differences in room temperature, lamp intensity, dithiothreitol strength, and hRXRα($D_\beta E$) concentration in stock solutions), data were converted to % quench at 337 nm using Equation 4.1:

$$Q=100\left(1-\frac{\Theta}{\Theta_0}\right) \tag{4.1}$$

where Θ is the signal intensity, Θ_0 is the signal intensity in the absence of ligand, and Q is % quench. The quench values were then nonlinearly fit to Equation 4.8, which is derived by substituting Equations 4.2 and 4.3 into Equation 4.5 and solving for $[R*L]$ (Equation 4.6), then substituting the result into Equation 4.7:

$$[R]=[R_{free}]+[R*L] \tag{4.2}$$

$$[L]=[L_{free}]+[R*L] \tag{4.3}$$

$$R*L \rightleftharpoons R_{free}+L_{free} \tag{4.4}$$

$$K_d=\frac{[R_{free}][L_{free}]}{[R*L]} \tag{4.5}$$

$$[R*L]=\frac{[R]+[L]+K_d-\sqrt{([R]+[L]+K_d)^2-4[R][L]}}{2} \tag{4.6}$$

$$Q=M\left(\frac{[R*L]}{[R]}\right) \tag{4.7}$$

$$Q=M\left(\frac{[R]+[L]+K_d-\sqrt{([R]+[L]+K_d)^2-4[R][L]}}{2[R]}\right) \tag{4.8}$$

where $[R*L]$ is the concentration of receptor-ligand complex; $[R]$ is the total receptor concentration; $[R_{free}]$ is the concentration of unbound receptor; $[L]$ is the total ligand concentration; $[L_{free}]$ is the concentration of unbound ligand; and M is the maximum quench at saturation of receptor sites.

4.2.3 Self-Association

Analytical ultracentrifugation (AUC) was performed on a Beckman Coulter (Fullerton, CA) XL-A in equilibrium mode using a 6-channel An-50 Ti rotor. Samples were initially prepared as approximately 30 µM stocks by exchanging the storage buffer for 100 mM NaCl and 10 mM NaP_i (pH 7.0) using at least three rounds of exchange in Millipore (Billerica, CA) Microcon devices to a minimum 1,000x dilution of storage buffer components. (In our hands, hRXRα($D_\beta E$) undergoing dialysis tended to precipitate unless the buffer contained approximately 10% or greater glycerol.) Denaturing runs also included 6.0 M guanidinium chloride (GdmCl) in the buffer. Three concentrations where run simultaneously as 1x, 2x, and 4x dilutions. Samples were initially run at 3,000 xg and scanned to ensure even distribution throughout the cells, then sequentially run to 12,000 xg, 16,000 xg, and 20,000 xg for 15 hours each, with a minimum of 4 scans at one hour intervals to ensure equilibrium at each stage.

Data were analyzed according to the following equations (298):

$$C_{1,r}=C_{1,r_0}\mathrm{e}^{M\frac{(1-\bar{\nu}\rho)\omega^2}{2RT}(r^2-r_0^2)} \tag{4.9}$$

$$C_{n,r}=K_{x,n}C_{1,r}^n \tag{4.10}$$

$$\Theta=\Theta_0+\sum_{i=1}^{x} C_{i,r}\epsilon_{280}b\,i \tag{4.11}$$

where $C_{1,r}$ is the concentration of monomer at radial position r; C_{1,r_0} is the concentration of monomer at the reference radial position r_0 (arbitrarily chosen as the radial position of the first data point being fit); M is the molecular mass of the monomer; $\bar{\nu}$ is the partial specific volume; ρ is the solution density; ω is the rotational velocity in radians sec^{-1}; R is the gas constant; T is temperature in Kelvin; $C_{n,r}$ is the concentration of the oligomeric species containing n monomer units; $K_{x,n}$ is the equilibrium

constant for the transition from species x (monomer or dimer) to species n (dimer or tetramer); Θ_0 is the baseline offset; and b is the path length (1.2 cm).

A nonlinear algorithm was used to fit Equation 4.11, making use of the measured noise for each data point. The range of data fit for each cell was from the first reliable data point after the meniscus (generally 0.015 cm later) to 0.01 cm before the bottom of the cell or the first data point with an absorbance intensity of > 1.5, whichever came first. Errors estimates are 1 standard deviation.

Baseline values for the fits were determined nonlinearly using the average absorbances of the 3,000 xg scans, the known ratios of concentrations loaded into the cells, and an additional "data point" to minimize the total deviation from zero. Baselines were then held fixed for all further analysis. As others have noted (299), allowing the baseline values to float as part of the fit leads to very high correlations in the parameters, and occasionally results in physically unrealistic parameters. Runs were analyzed individually and globally as described in Section 4.3.2. Partial specific density of hRXRα($D_\beta E$) was estimated to be 0.7388 ml mg^{-1} (native) or 0.7277 ml mg^{-1} (denatured) using the program SEDNTERP version 1.09 (300). Buffer densities were also estimated using SEDNTERP, and confirmed to match the theoretical values within experimental error using an Anton-Paar (Graz, Austria) DMA-60 densimeter with a DMA-512 remote cell. Density values used were 1.00351 mg mL^{-1} for native buffer, and 1.14755 mg mL^{-1} for denaturing buffer.

4.2.4 Thermal Denaturation

Circular dichroism spectroscopy (CD) was performed on a Jasco (Easton, MD) J-810 spectropolarimeter equipped with a peltier temperature controller and six-cell changer, using 0.1 cm cuvettes (300 µL volume).

Unless otherwise noted, conditions were 10 μM hRXRα($D_\beta E$) in 100 mM NaCl, 10 mM NaP_i, pH 7.0. Full scans were performed from 250-200 nm at 5 °C and 250-205 nm at 85 °C using a scan rate of 50 nm min^{-1}, 4 accumulations, 1 sec response, 1 nm data interval, and 1 nm slit width. All scans were baseline subtracted using a blank scan (containing everything except hRXRα($D_\beta E$)) at 5 °C. Thermal ramps were performed from 5-85 °C at 1 °C min^{-1} unless noted, 222 nm, 5 nm slit width, 1 sec response, and a data interval of 1 °C. Samples containing ligand were kept to a maximum of 1% organic solvent. To correct for any solvent effects, scans and ramps were also performed for buffers containing 1% organic solvent but no ligand, and used as the comparison point for ligand effects. Although a small (< 0.5 °C) decrease in T_m was observed due to solvent alone, the shift was not statistically significant.

Data were nonlinearly fit to a simple sigmoidal model allowing for sloped upper and lower baselines (Equation 4.16), derived by substituting Equations 4.12, 4.13, and 4.14 into Equation 4.15 and rearranging:

$$\Theta_N = B_N + m_N T \tag{4.12}$$

$$\Theta_D = B_D + m_D T \tag{4.13}$$

$$F_D = \frac{[D]}{[R]} = \frac{1}{1+e^{m(T-T_m)}} \tag{4.14}$$

$$\Theta = \Theta_D F_D + \Theta_N (1 - F_D) \tag{4.15}$$

$$\Theta = \Theta_N + \frac{\Theta_D - \Theta_N}{1+e^{m(T-T_m)}} \tag{4.16}$$

where Θ_N and Θ_D are the native and denatured signals as a fixed value plus a temperature dependence; F_D is the fraction of the total receptor concentration that is denatured (defined as a sigmoidal curve); Θ is the measured signal; T is the temperature in °C; m is essentially the "slope" of

the transition and reflects the cooperativity; and T_m is the midpoint of the transition. Fits used at least three normalized replicates; normalized plots were used to reduce the dependency of the calculated T_m on the exact values for Θ_N and Θ_D, which significantly decreased error estimates without influencing the center value.

Differential scanning calorimetry (DSC) was performed using the same scan rate and buffer conditions as in the CD spectroscopy. However, due to the lower sensitivity of the Calorimetry Sciences Corporation (New Castle, DE) N-DSCII used, a standard concentration of 30 µM hRXRα($D_\beta E$) was used unless noted. Ramps were run from 5-105 °C, with 10 minute equilibration times at each extreme. A blank cycle was run before each DSC scan, and the hRXRα($D_\beta E$) added during the blank cooling cycle, when the temperature was 20-25 °C. T_m estimates were taken as the midpoint of the most intense peak of the DSC trace after correcting for baseline.

4.2.5 Chemical Denaturation

Chemical denaturation was performed using CD spectroscopy, essentially as described in section 4.3.3. However, scans were taken only at 25 °C instead of heating, and from 250-212 nm due to guanidinium chloride absorbance. Buffer conditions were identical, except with increasing concentrations of GdmCl. Each measurement was individually baseline subtracted, and cuvettes cleaned between measurements. Measurements were performed using both 10 µM and 2.0 µM hRXRα($D_\beta E$). Fluorescence denaturation experiments were performed identically, except that 1.0 cm cuvettes were used with a total volume of 1,000 µL and 1.0 µM hRXRα($D_\beta E$), using a slit width of 3 nm, 300-380 nm scan range, 1 nm resolution, and 1 sec integration time. To show reversibility, GdmCl was removed by repeated buffer exchange using Millipore Micron devices, and the final spectrum corrected for loss of protein based on absorbance (typically about 10%).

Only the 222 nm CD or 334 nm fluorescence data were fit.

Data were fit to a model allowing for tetramer, dimer, monomer, and unfolded species, derived following previous approaches (301, 302). In short, given the relationships in Equations 4.17 through 4.22, data were fit to Equation 4.23:

$$[R]=4[T]+2[D]+[M]+[U] \tag{4.17}$$

$$K_{D,T}=\frac{[T]}{[D]^2} \tag{4.18}$$

$$K_{M,D}=\frac{[D]}{[M]^2} \tag{4.19}$$

$$K_{U,M}=\frac{[M]}{[U]} \tag{4.20}$$

$$K=e^{\frac{-\Delta G}{RT}}=e^{\frac{-m([Den]-C_M)}{RT}}=e^{\frac{-m[Den]+\Delta G_{D,H_2O}}{RT}} \tag{4.21}$$

$$[\Theta]=4\frac{[T]}{[R]}\Theta_T+2\frac{[D]}{[R]}\Theta_D+\frac{[M]}{[R]}\Theta_M+\frac{[U]}{[R]}\Theta_U \tag{4.22}$$

$$[\Theta]=4K_{D,T}K_{M,D}^2\frac{[M]^4}{[R]}\Theta_M+2K_{M,D}\frac{[M]^2}{[R]}\Theta_D+\frac{[M]}{[R]}\Theta_M+\frac{[M]}{K_{U,M}[R]}\Theta_U \tag{4.23}$$

where *[R]*, *[T]*, *[D]*, *[M]*, and *[U]* are the total concentrations of receptor, tetramer, dimer, monomer, and unfolded species, respectively; R is the universal gas constant; T is temperature in Kelvin; *[Den]* is the concentration of denaturant; m and ΔG_{H_2O} are the slope and intercept, respectively, of a plot of ΔG_D versus *[Den]*; and C_m is the *[Den]* at which the two species have equal concentration. The implicit assumption that ΔG_D is linearly dependent on *[Den]* has been previously validated (303, 304). Fits were performed using a nonlinear algorithm as described below, using CD data converted to molar ellipticity (deg cm^2 $dmol^{-1}$) and raw fluorescence intensities weighted to provide equal impact on the fit with the CD data.

Baselines were fixed at zero based on prior work (269, 305) and lack of evidence for any slope in the apparent upper and lower baselines.

4.2.6 Data Analysis

All nonlinear fits used a variation of the Nelder-Mead simplex algorithm (306) for least-squares minimization; see Chapter 6 for a more detailed presentation. When replicate data sets were available for global fits (AUC, thermal denaturation, and chemical denaturation), a global fit was used to increase confidence in, and reduce correlations between, parameters (307-310). Quality of minimization was determined using χ^2. During global fits, individual data sets were also weighted for the number of data points and possibly an arbitrary factor to correct for large differences in the magnitude of the data. χ^2 is only a relative measure of a fit, so while smaller is better during the fitting process for a particular data set, χ^2 values for different data sets are not comparable. In addition, each fit was analyzed using a runs test, which is a measure of the randomness in the residuals of the fit. The parameter Z should be close to zero for a random distribution; an absolute difference from zero of approximately 2 or greater indicates significant trends in the residuals, which in turn suggests the model may not be appropriate for the data (311). Errors were estimated as described in Section 6.2. In short, each parameter was individually displaced from the best-fit value and held fixed while a new best fit was obtained; when the resulting fit was statistically different from the best fit, the displacement was taken as the error. This method tends to produce larger, but more reasonable, error estimates than traditional methods such as matrix inversion (308, 312-314). The reported errors are always ±1 standard deviation.

Linear fits were performed using standard linear regression. For linear correlations involving lower limit K_d values, a reasonable estimate was

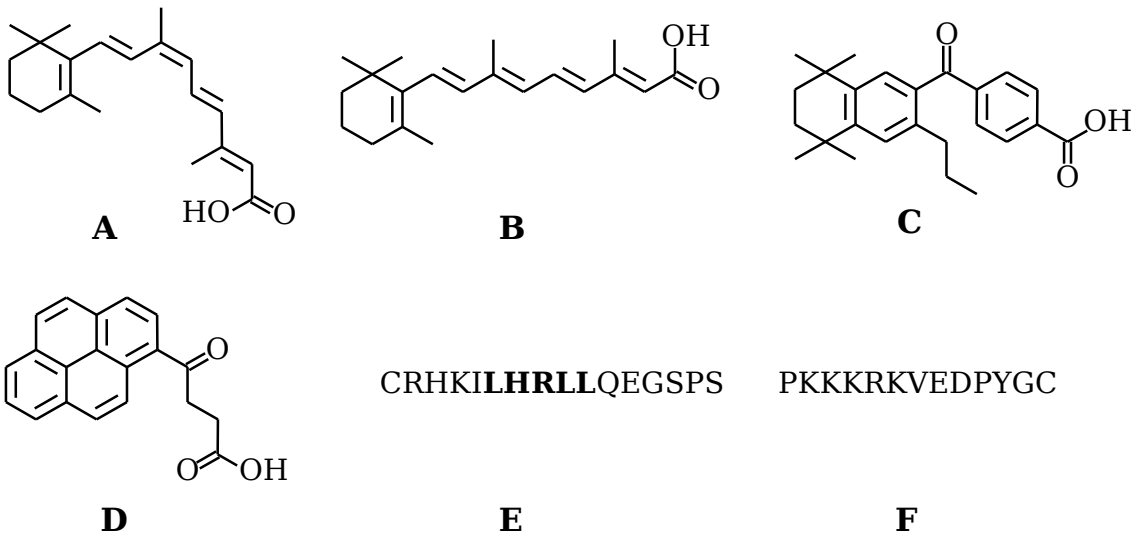

Figure 4.1. Ligands and peptides. (A) 9-*cis* retinoid acid (9cRA). (B) All-*trans* retinoic acid (atRA). (C) LG335. (D) γ-oxo-1-pyrenebutyric acid (OPBA). (E) LxxLL peptide (LxxLL motif highlighted), from steroid receptor coactivator-1. (F) nonsense peptide.

set as 1.5-fold greater than the K_d lower limit, with errors set at 33% (i.e., the lower limit to twice the lower limit). This range spans the K_d values associated with the entirely physically realistic range of maximum quench values, approximately 50-95%, as observed during numerous titrations.

All statistical significant was measured at the $p \leq 0.5$ level (two standard deviations when comparing values).

4.3 Results

4.3.1 Ligand Binding

Binding constants of hRXRα($D_{\beta}E$) for various ligands (see Figure 4.1) were determined in fluorescence assays. 9cRA is a known activator of hRXRα (176), LG335 weakly activates hRXRα (219, 315), and atRA has not been shown to activate hRXRα to any measurable extent, as small amounts of activation are generally attributed to formation of small amounts of 9cRA in the cell (176). OPBA has been used as a target ligand for engineered hRXRα in the Doyle laboratory; it weakly activates hRXRα at concentrations

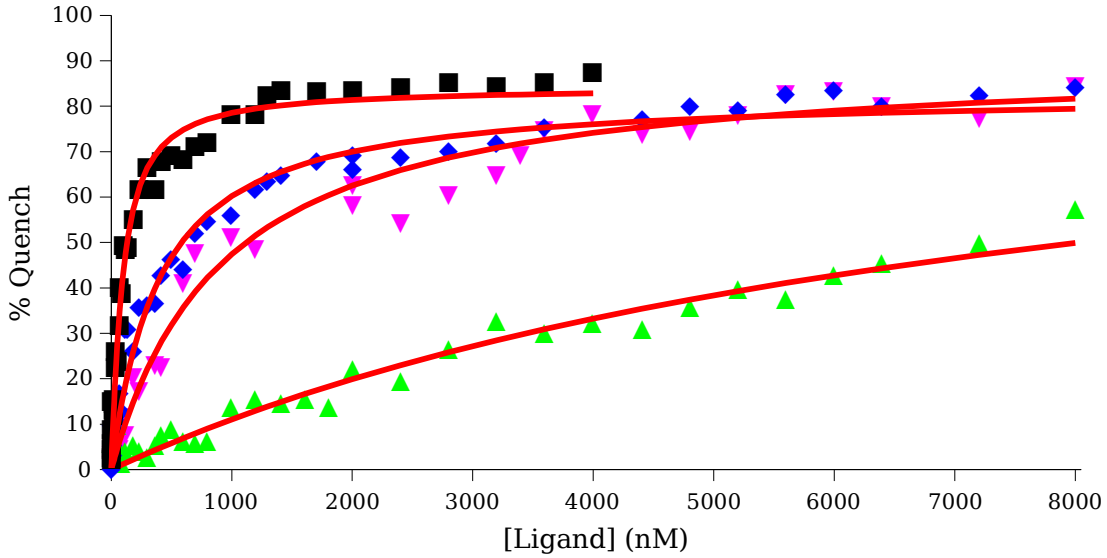

Figure 4.2. Binding data and fits for 50 nM hRXRα(DβE) with ligands. (Black box, ■) 9cRA. (Blue diamond, ◆) LG335. (Green up triangle, ▲) OPBA. (Magenta down triangle, ▼) atRA. (Red lines, ▬) Best fit curves. The upper limit of data collection is 8000 nM or at least 40×K_d.

greater than 10 μM (221), beyond the concentration range at which the other ligands can be assayed, due to limited solubility. The binding pocket of hRXRα(DβE) contains two tryptophan residues, the emissions from which are reduced (quenched) upon ligand binding. The assays were modified from previous work (247, 249) due to signal loss over time, which leads to "tryptophan quenching" indistinguishable from that caused by ligand binding (see Figures 4.2 and 4.3). Moreover, the effect was general, as the same relative signal loss was observed even when starting with saturated receptor. Our method of individual measurements in clean cuvettes to minimize signal loss provided reliable, reproducible data. Attempts to minimize loss using coatings on the cuvettes or alternative buffer conditions reduced the signal loss somewhat, but did not eliminate it as a significant factor over the course of a titration. Figure 4.2 and Table 4.3 show the quench data, fits, and calculated K_d values for each ligand. Due to the relatively low solubility of OPBA and low affinity for hRXRα(DβE), we are

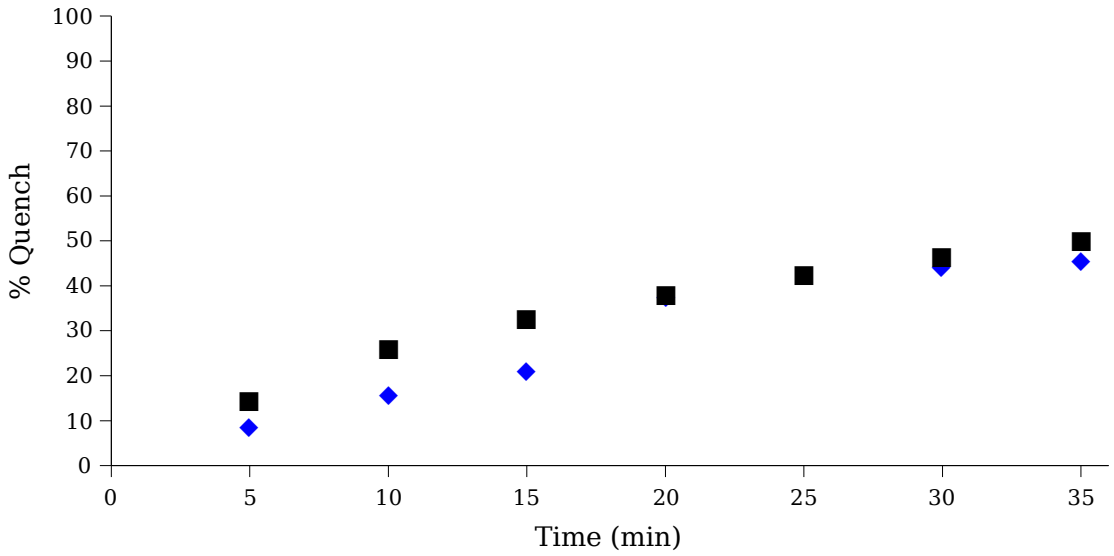

Figure 4.3. Signal loss over time for 50 nM hRXRα($D_\beta E$). (Black box, ■) Receptor alone. (Blue diamond, ◆) Receptor with 1 µM 9cRA.

able to only put a lower limit on the K_d by assuming a reasonable maximum quench (approximately halfway between the observed maximum quench and the lowest maximum quench from the fits for other ligands). The upper limit of data collection for each set was either 8000 nM ligand or at least 40-fold greater than the K_d, whichever came first. The 8000 nM cutoff was used because ligands began to precipitate in the assay buffer in the 9-12 µM range. At concentrations greater than $40 \times K_d$, additional data points do not significantly influence the fit.

Due to the weak intrinsic fluorescence signal and the rate of signal loss, assays with receptor concentrations below 50 nM were unreliable. As the K_d values for the ligands are of comparable affinity (i.e., less than 100-fold greater), the hRXRα($D_\beta E$) concentration must be included as an explicit parameter when fitting the data. However, our analysis of simple models suggests that as much as a two-fold error in the hRXRα($D_\beta E$) concentration used in the fit does not shift the resulting K_d estimate outside the error range. This concentration tolerance is beneficial, as it eliminates possible

Table 4.3. Ligand binding parameters for hRXRα($D_\beta E$).

	Ligand alone		Ligand + LxxLL peptide	
Ligand	**K_d (nM)**	**Max Quench (%)**	**K_d (nM)**	**Max Quench (%)**
9cRA	70 ± 20	84 ± 4	35 ± 14	75 ± 4
LG335	370 ± 100	83 ± 5	110 ± 30	71 ± 3
atRA	900 ± 300	91 ± 5	520 ± 160	87 ± 5
OPBA	> 3800	65[α]	> 3800	55[α]

[α] Fixed as described in text.

complexities arising from fitting different models, such as dimers with only a single active binding site. On the other hand, we are also unable to comment on possible cooperative (positive or negative) behavior during binding, as a change in receptor concentration is equivalent to a change in the number of ligand binding sites. Therefore, no information regarding the oligomericity hRXRα($D_\beta E$) can be obtained from binding studies alone.

The coactivator LxxLL peptide used is from the steroid receptor coactivator-1 with an extra C-terminal cysteine (C-^{688}SRC-1^{702}, see Figure 4.1), and is similar to that used in previous studies (276, 283, 295). The peptide had no effect on the tryptophan emission up to concentrations of 100 μM, and so we were unable to obtain a direct binding affinity for the peptide using this technique. However, previous reports suggest the K_d is at most 6 μM in the absence of ligand and 2 μM in the presence of 9cRA, and possibly 10-fold lower (276, 295). Therefore, a concentration of 20 μM LxxLL peptide should result in at least 90% of the hRXRα($D_\beta E$) being bound to the peptide. To observe the effect of the LxxLL peptide on the ligand K_d, the ligand binding assays were repeated using a fixed concentration of 20 μM LxxLL peptide in the buffer. Figure 4.4 and Table 4.3 show the resulting data and K_d values. No further shift in the K_d values occurred at concentrations of LxxLL peptide up to 100 μM, suggesting that hRXRα($D_\beta E$)

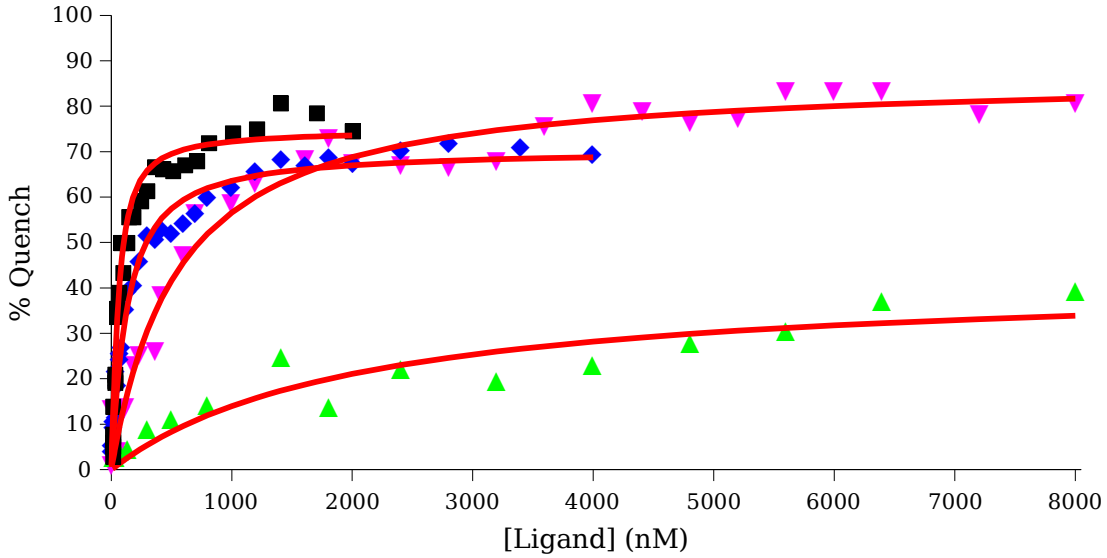

Figure 4.4. Binding data and fits for 50 nM hRXRα(DβE)with ligands in the presence of 20μM LxxLL peptide. (Black box, ■) 9cRA. (Blue diamond, ◆) LG335. (Green up triangle, ▲) OPBA. (Magenta down triangle, ▼) atRA. (Red lines, ▬) Best fit curves. The upper limit of data collection is 8000 nM or at least 40×K_d.

is indeed predominantly bound to the LxxLL peptide at 20 μM. The use of a nonsense peptide did not produce a shift in the observed K_d for 9cRA (70 ± 20 nM and maximum quench of 83 ± 4%), indicating that the LxxLL peptide had a specific effect on the ligand affinity.

4.3.2 Self-Association

To determine the self-association of the natively folded receptors, equilibrium analytical ultracentrifugation was performed at three protein concentrations and three spin velocities. During the experiments, the three protein concentrations were in separate, stacked cells, so also had different radial positions. Figure 4.5 shows data at 12,000 rpm fit to a single species model (floating molecular mass), as well as the expected curves for pure monomer, dimer, and tetramer species. The obvious mismatch between the data and the expected single species curves indicates that the samples were either a single species with a molecular mass between dimer and tetramer,

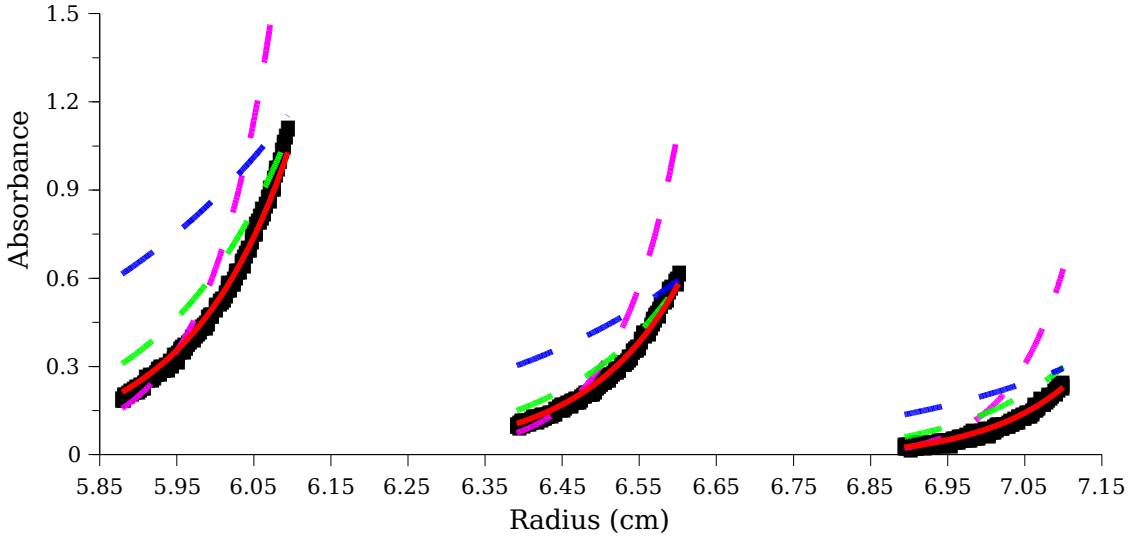

Figure 4.5. Representative hRXRα(DβE) AUC data fit to a single species model with theoretical curves for monomeric, dimeric, and tetrameric species. (Black box, ■) 12,000 rpm data. (Red lines, ▬) Best fit curves for single species model with floating molecular weight. (Blue lines, ▬▬) Expected curves for pure monomer. (Green lines, ▬▬) Expected curves for pure dimer. (Magenta lines, ▬▬) Expected curves for pure tetramer.

or a mixture of species. The residuals for the floating molecular mass model, shown in Figure 4.6, are clearly not random, as indicated by the tendency of the residuals to cluster in positive or negative groups. These trends suggest that a mixture is the more likely explanation for the data. Moreover, fits to individual runs and single cells (i.e., single concentration at multiple speeds) clearly indicate an increase in molecular mass with increasing concentration, suggesting an equilibrium of states. The estimated molecular mass for a single species, 78.3 ± 1.5 kDa, rules out the possibility of a monomer-dimer equilibrium, as the monomer expected mass is 29.1 kDa; analysis of data for hRXRα(DβE) under identical conditions except for the addition of 6 M GdmCl produces an average molecular mass of 32.3 ± 0.6 kDa. Monomer-tetramer dimer-tetramer non-equilibrium (fixed ratio) models are not improvements over the single species model, whereas a dimer-tetramer equilibrium model is significantly better. Finally,

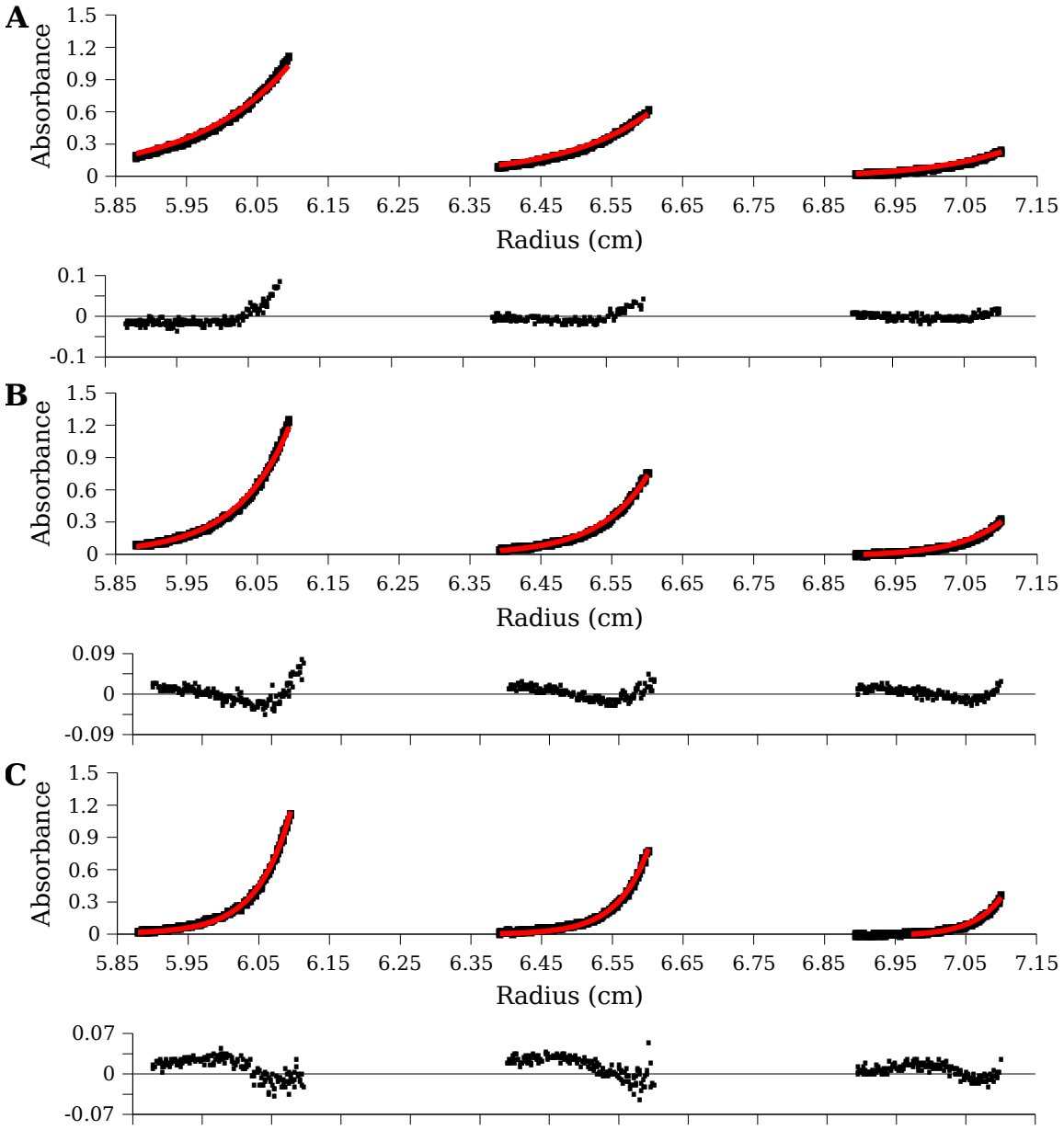

Figure 4.6. hRXRα(DβE) AUC data fit to a single species model with residuals. (A) 12,000 rpm. (B) 16,000 rpm. (C) 20,000 rpm. (Red lines, ▬) Best fit curves. Vertical scale changes on residual plots to more clearly show trends in the residuals.

as shown in Figure 4.7 and Table 4.4, a monomer-dimer-tetramer model produces a greatly improved fit.

Two related problems suggest that additional processes also occurred during the ultracentrifugation. First, the apparent average mass of

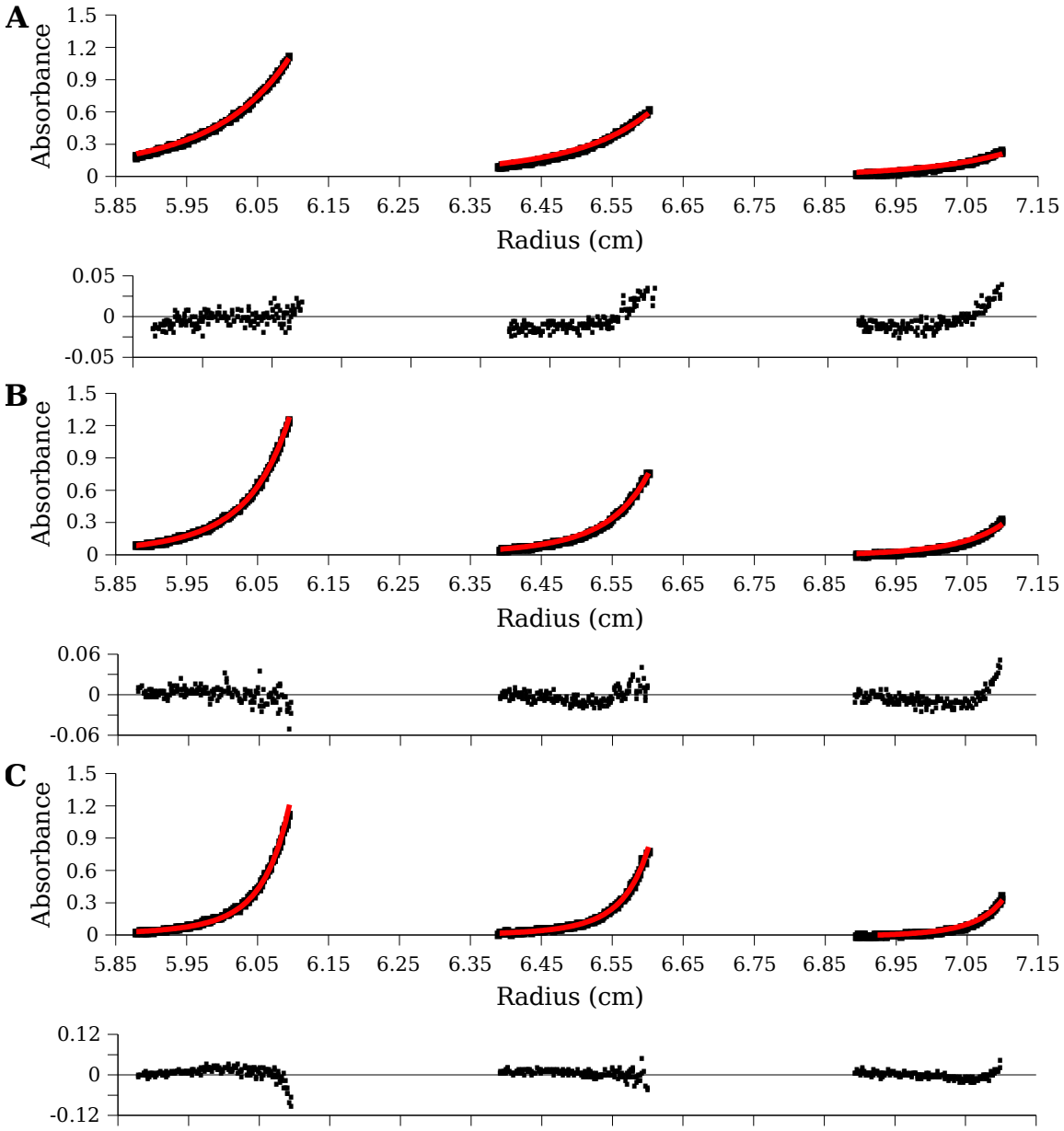

Figure 4.7. hRXRα($D_{\beta}E$) AUC data fit to a monomer-dimer-tetramer model with residuals. (A) 12,000 rpm. (B) 16,000 rpm. (C) 20,000 rpm. (Red lines, ▬) Best fit curves. Vertical scale changes on residual plots to more clearly show trends in the residuals.

hRXRα($D_{\beta}E$) decreases with increasing spin velocity when the nine data sets are analyzed in three sets grouped by spin velocity or individually. This decrease suggests a loss of material and/or non-specific aggregation. Second, as can be seen in Figure 4.8, a dimer-tetramer model gives a high

Table 4.4. Global fit parameters of all AUC data for hRXRα($D_{\beta}E$).

	Fit Quality[α]		
Model[β]	**χ^2**	**Z**	**kDa or K_d[β]**
Single species	48.3	19.5	78.3 ± 1.5 kDa
4M ⇌ T	55.0	13.6	1.6 ± 0.4 μM^3
2D ⇌ T	30.4	8.6	35 ± 9 μM
4M ⇌ 2D ⇌ T	27.2	6.5	1.55 ± 0.07 μM (2M ⇌ D) 12.9 ± 0.9 μM (2D ⇌ T)

[α] See Chapter 6 for details.
[β] M = Monomer, D = Dimer, T = Tetramer.

quality fit to the 12,000 rpm velocity data, which is not further improved by introduction of a monomeric species (see Table 4.5). Fits to the 16,000 and 20,000 rpm data alone are much poorer (data not shown), even after introduction of a monomeric species. The fits are qualitatively similar, but fits to the 12,000 rpm data alone have much higher statistically reliability. Although the aggregation was apparently greatly reduced in buffer containing 6 M GdmCl, some spin velocity dependence is still apparent in the fits; using only the 12,000 rpm data produces a superior fit and an estimated molecular mass of 29.7 ± 1.2 kDa, which is in good agreement with the expected molecular mass (29.1 kDa). Therefore, we ultimately used only the 12,000 rpm data, at all three concentrations, to avoid the influence of non-specific effects. Use of a single spin velocity prevents use of mass constraints (299, 316, 317) to further refine the fits, but imposing mass constraints is not justified anyway due to an apparent loss of material as a function at higher spin velocities (decreased total absorbance). The total absorbance decrease was reversible, as performing a second run on the same samples (after halting the ultracentrifuge and mixing) reproduced the first runs quite closely. This behavior suggests that the non-specific behavior arose due to the centrifugation, and is not due to degradation of

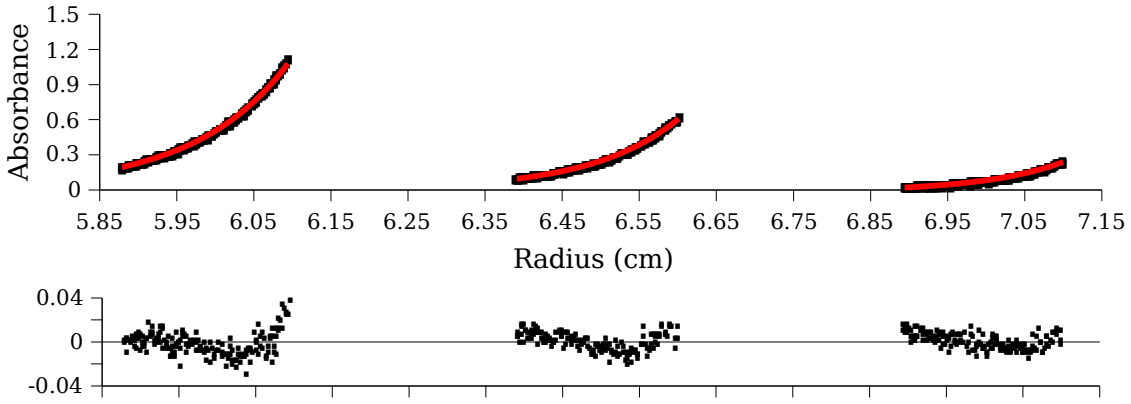

Figure 4.8. hRXRα(DβE) 12,000 rpm AUC data fit to a dimer-tetramer model with residuals. (Red lines, ▬) Best fit curves.

the sample during the long runs. As the residuals from the 12,000 rpm data alone are still not entirely random (Z score of 2.5, slightly higher than the target of less than 2), a little non-specific aggregation may have occurred even at that spin velocity; this small effect does not significantly impact our analysis.

Native polyacrylamide gels produced results with considerable ambiguity, and we were unable to reliably determine absolute molecular masses. Moreover, significant aggregation occurred under some conditions. Two bands were always observed, with an intensity ratio in approximate agreement with the AUC results for the dimer-tetramer model.

Due to limited ligand solubility (approximately 10 μM) and the relatively high protein concentrations required for AUC (5-30 μM), attempts to determine ligand effects on the oligomerization were unsuccessful. The precipitation of ligands prevented preparation of reference cells containing an equimolar concentration of ligand for baseline subtraction; since the ligands all have absorbance at 280 nm comparable to hRXRα(DβE), the differences between the reference and samples cells would be significant. In addition, our observations indicate that the molar absorptivity of 9cRA in particular is quite sensitive to environment (e.g., the molar absorptivity in

Table 4.5. 12,000 rpm AUC global fit parameters for hRXRα($D_\beta E$).

	Fit Quality[α]		
Model[β]	**χ^2**	**Z**	**K_d[β]**
4M ⇌ T	14.9	15.4	0.9 ± 0.4 μM[3]
2D ⇌ T	3.6	2.5	20 ± 2 μM
4M ⇌ 2D ⇌ T	3.7	2.6	0.242 ± 0.001 nM (M ⇌ D) 20 ± 3 μM (D ⇌ T)

[α] See Chapter 6 for details.
[β] M = Monomer, D = Dimer, T = Tetramer.

water is less than half the molar absorptivity in ethanol), and so we would have had to introduce the 9cRA molar absorptivity as a floating parameter in fits. Addition of this parameter would introduce large uncertainties into the final parameter estimates due to strong cross-correlations.

4.3.3 Thermal Denaturation

To probe the oligomeric and unfolding energetics of hRXRα($D_\beta E$), we turned to thermal denaturation studies. However, hRXRα($D_\beta E$) irreversibly denatures when heated, so we are unable to extract true thermodynamic information from the data. Shown in Figure 4.9 are CD spectra of the same sample at 5 °C, after heating to 85 °C, and after cooling back to 5 °C. The 5 °C spectrum indicates the expected predominantly α-helical structure for hRXRα($D_\beta E$) (the two negative peaks at 222 and 208 nm). The 85 °C spectrum is suggestive of a predominantly β-sheet structure (a single, broad negative peak around 216 nm) rather than an unfolded structure, and the spectrum after cooling is not significantly different. The rate of cooling had no effect on the resulting spectrum at 5 °C. As the small change in the CD spectrum upon cooling was linked to a much stronger change in the absorbance, it is probably not indicative of a significant change in structure. Single wavelength spectra of hRXRα($D_\beta E$), shown in Figure 4.10, have clear

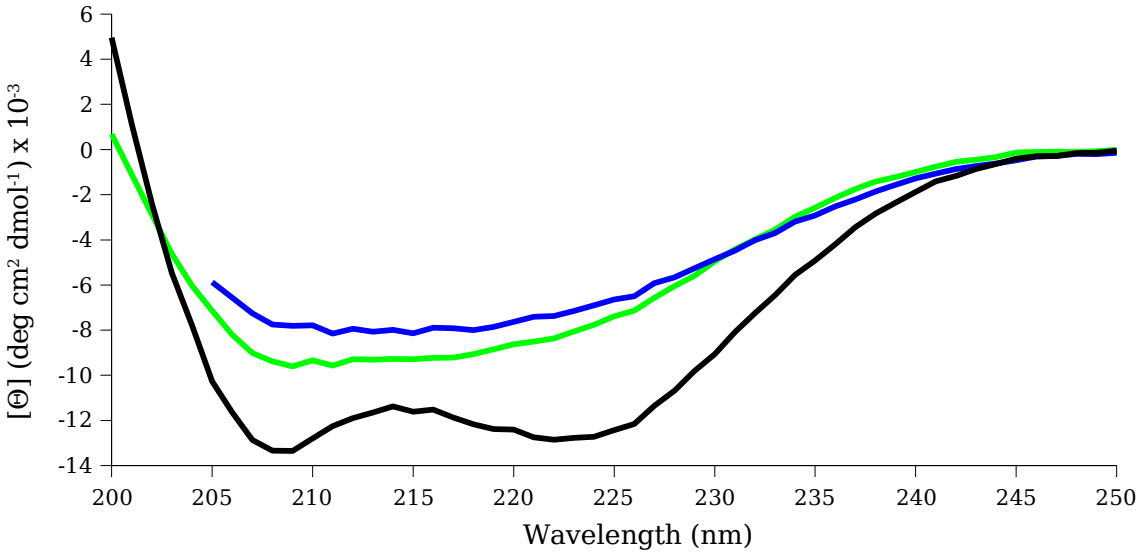

Figure 4.9. Far-UV CD spectra of 10 µM hRXRα($D_{\beta}E$). (Black line, ▬) 5 °C. (Blue line, ▬) 85 °C. (Green line, ▬) 5 °C after slowly cooling from 85 °C.

transitions during heating, but no significant transition during cooling. hRXRα($D_{\beta}E$) had identical behavior in DSC experiments (Figure 4.11), where a transition during the first heating was not followed by a transition while cooling or upon reheating. Some DSC runs also had a significant "tail" on the DSC transition, which is suggestive of aggregation. This aggregation is visually apparent as a thin film that coated cuvette surfaces. The protein did not precipitate in a traditional manner, and some portion remained in solution. The soluble portion of hRXRα($D_{\beta}E$) heated to 55 °C or 85 °C, rapid cooled, and analyzed by AUC. The soluble fraction had a significantly increased average molecular mass (average molecular masses of 109 ± 7 kDa and 207 ± 7 kDa, respectively). We performed no further characterization of this aggregate, other than to determine complete removal of the aggregate from cuvette surfaces required strong detergent (> 1%), heating to > 90 °C for at least 15 minutes, and acid washing (> 0.1 M HCl).

In addition to being irreversible, the transition exhibited scan-rate

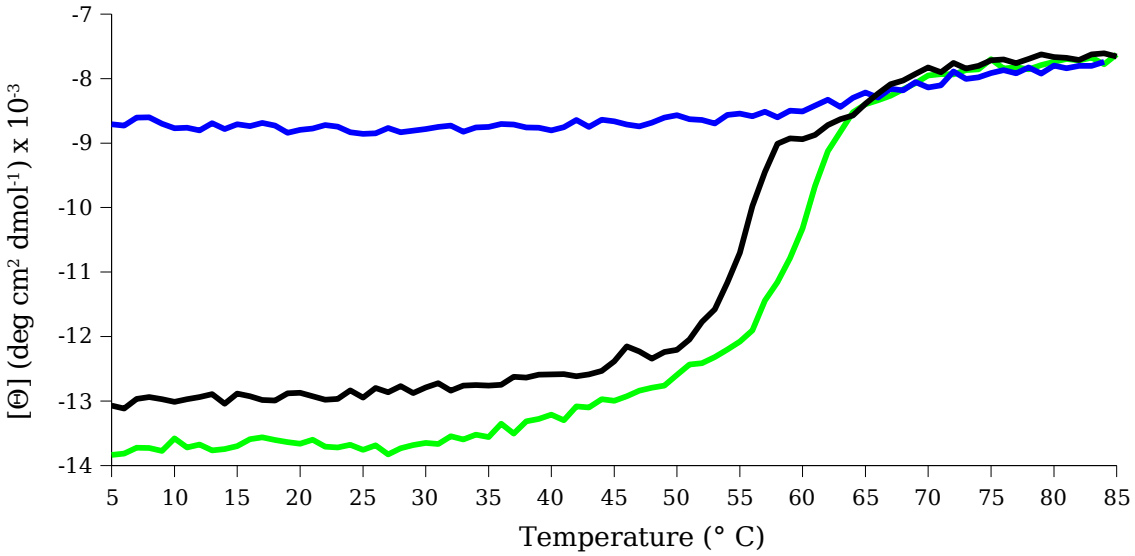

Figure 4.10. Temperature-dependent CD spectra of 10 µM hRXRα(DβE) at 222 nm. (Black line, ▬) Heating. (Blue line, ▬) Cooling. (Green line, ▬) Heating a second sample in the presence of equimolar 9cRA.

dependence. Increasing DSC scan rates caused an increasing apparent T_m, for an increase of nearly 3 °C as the scan rate was increased from 0.25 to 1.5 °C min^{-1}. Moreover, we did not observe an effect of protein concentration on the apparent T_m. The combination of these effects, the fact that even partial heating is irreversible, and that an "unfolded" state was never clearly present, indicate that we cannot extract thermodynamic information from the data (318, 319). Therefore, given the basic agreement between CD and DSC data, all further experiments were done using fixed conditions via CD spectroscopy to minimize material consumption. Although the CD spectrum upon heating is suggestive of two transitions (Figure 4.10), and the corresponding DSC trace clearly contains two transitions (Figure 4.11), all data was fit to a model for a single, irreversible, transition to determine the apparent T_m. We justify this approach on the basis of three observations: (i) the relative intensities of the two transitions varied considerably between runs and exhibited strong dependence on buffer

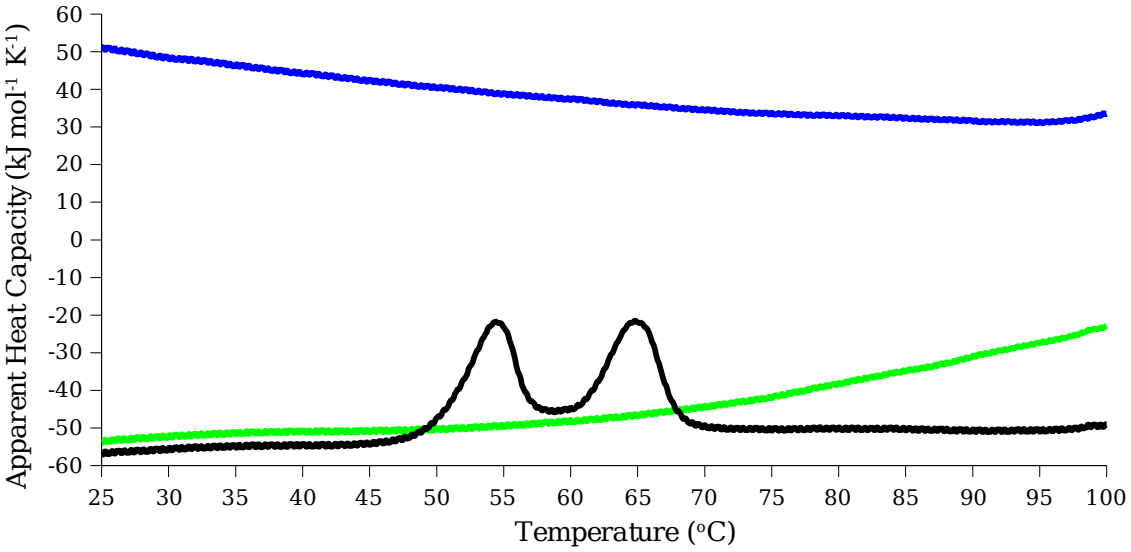

Figure 4.11. DSC traces of 30 µM hRXRα($D_{\beta}E$). (Black line, ▬) Initial heating trace. (Blue line, ▬) Cooling trace. (Green line, ▬) Second heating trace.

conditions; (ii) because the first transition is much stronger than the second in the CD spectra, so the estimated T_m values from single transition fits are essentially identical to the lower T_m values from multiple-transition models; and (iii) because the dependencies (on other parameter values) in the estimated T_m values for two-transition fits are very large, producing great uncertainty in the values (large error estimates).

Despite the irreversible nature of the transition, relative T_m values of the native structure may still be compared when heated under different conditions (e.g., varying ionic strength). Ionic strength had only a small, statistically insignificant, increase on the apparent T_m, throughout a range 0 to 1 M NaCl (in a constant background of 10 mM NaP_i). A pH range of 5 to 9 produced no shift in T_m, although a pH of 3 or 11 reduced the T_m to approximately 30 °C. All conditions produce an apparently identical final aggregate, except when hRXRα($D_{\beta}E$) was completely denatured using guanidinium chloride (see below) before heating, or when using an acetate

Table 4.6. Effect of ligands and LxxLL peptide on thermal stability of hRXRα($D_\beta E$).

Ligand	T_m (°C)	ΔT_m (°C)[α]	+LxxLL ΔT_m (°C)[β]
None	55.0 ± 0.7	-	3.3 ± 2.3
9cRA	-	6.3 ± 0.8	6.8 ± 1.2
LG335	-	3.9 ± 0.7	5.8 ± 1.1
atRA	-	-0.4 ± 1.1	4.0 ± 1.0
OPBA	-	-1.2 ± 2.1	1.9 ± 1.3

[α] ΔT_m indicates the change in T_m due to the presence of ligand.
[β] ΔT_m indicates the change in T_m due to the presence of ligand and peptide.

buffer, which caused hRXRα($D_\beta E$) to precipitate rather than form a film.

Unlike buffer conditions, ligands at equimolar concentrations had considerable effect on the apparent T_m. As shown in Figure 4.10 for 9cRA and Table 4.6 for all ligands, some ligands considerably increased the T_m of hRXRα($D_\beta E$). The LxxLL peptide also increased the T_m somewhat. However, the effect of the LxxLL peptide with each ligand was not additive (i.e., the effect from the ligand and peptide together was not simply the sum of the individual effects); in general, the stronger the effect on T_m of the ligand alone, the smaller the additional effect of the LxxLL peptide. Although the limited solubility of the ligands prevented the use of excess concentrations, the ligand Kd (see Table 4.3) are low enough that the receptor was near saturation at equimolar concentrations. The LxxLL peptide was used at a 20 μM concentration to enable comparisons with the binding assays; higher concentrations did not cause significantly different results.

Two additional interactions influenced the magnitude of the error estimates in Table 4.6: (i) the simultaneous unfolding of the LxxLL peptide when the hRXRα($D_\beta E$) aggregates, which has a significant effect on the CD signal; and (ii) possible binding of OPBA to the aggregated hRXRα($D_\beta E$), suggested by exothermic peaks in DSC traces with OPBA present, which

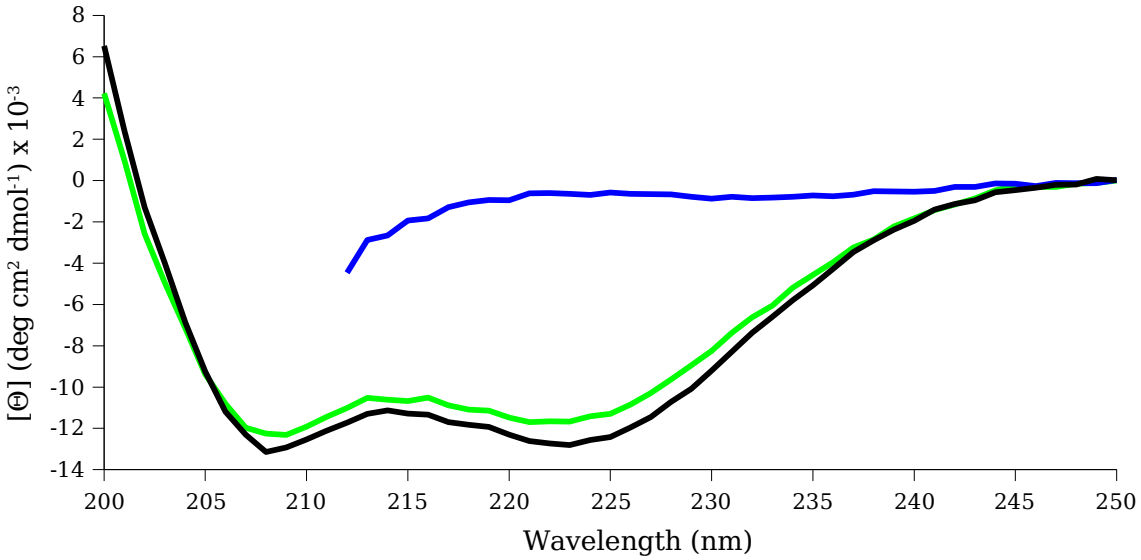

Figure 4.12. Effect of guanidinium chloride on CD spectra of 10 µM hRXRα(DβE). (Black line, ▬) 0 M GdmCl. (Blue line, ▬) 6 M GdmCl. (Green line, ▬) Following removal of the 6M GdmCl via repeated buffer exchange.

may influence the rate of formation of the aggregate. Both of these effects serve to increase the cross-correlation between *m* and the apparent T_m, which increases the uncertainties in the parameters.

4.3.4 Chemical Denaturation

Guanidinium chloride denatured hRXRα(DβE) reversibly, as shown in Figure 4.12. Despite the effect of pH on T_m noted above, at 25 °C hRXRα(DβE) was only partially unfolded throughout a pH range of 1.5 to 12.5, so only GdmCl was used for chemical denaturation experiments. Unfolding by GdmCl and establishment of a new equilibrium apparently occurred rapidly, as measurements taken 60 seconds after addition of hRXRα(DβE) to a solution containing GdmCl were identical to scans taken after more than hour. Similarly, 0.05% SDS was sufficient to completely denature hRXRα(DβE) within two minutes even in the absence of heating.

Figures 4.13 and 4.14 show the denaturation data in the absence and presence of 9cRA. Due to the oligomeric nature of hRXRα(DβE), the CD

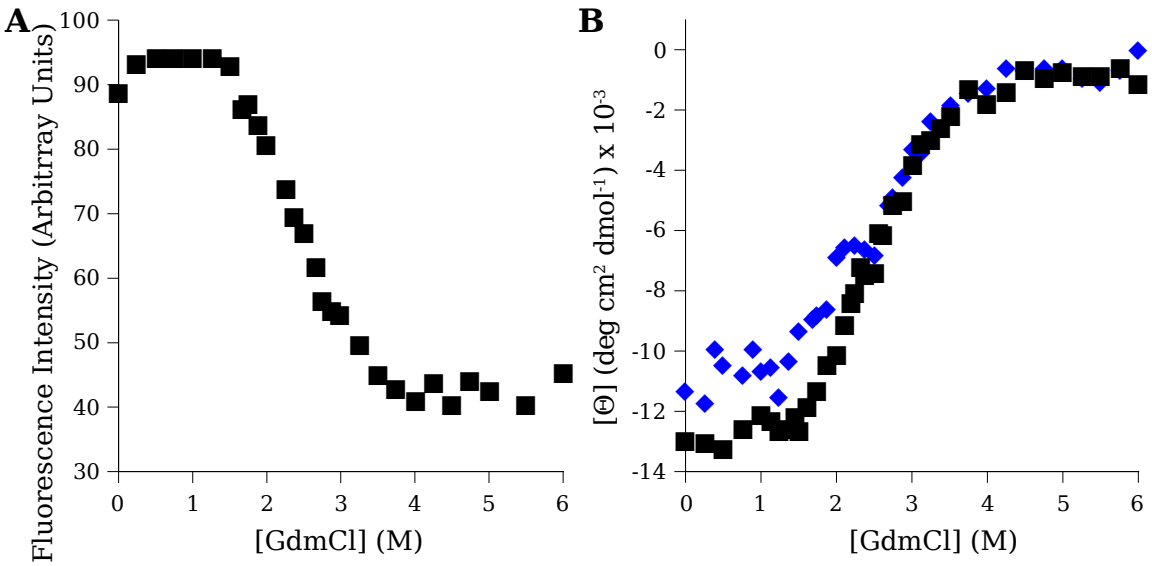

Figure 4.13. Guanidinium chloride induced unfolding of hRXRα($D_{\beta}E$). (A) Fluorescence-monitored unfolding (334 nm) using 1 μM receptor. (B) CD-monitored unfolding (222 nm), using 10 μM (black box, ■) and 2 μM (blue diamond, ◆) receptor.

spectra contain at least three distinct transitions in the absence of 9cRA. These states are not well separated, which leads to high correlations between parameters. To try to reduce the correlations, unfolding was measured at three concentrations (1.0, 2.0, and 10 μM), using two spectroscopic techniques (CD and fluorescence), and coupled to the dimer-tetramer equilibrium data available from AUC experiments. The fluorescence intensity decrease correlates with the expected shift in the maximum emission from 334 nm to 350 nm (320-322). However, rigorous analysis indicates that the data can be fit by a variety of different but statistically equivalent parameter sets; the basic problem is lack of resolution for individual states, not insufficient data points. Quantitative conclusions cannot be drawn from the data with additional measurements using alternative techniques such as light scattering or analytical ultracentrifugation in similar buffers.

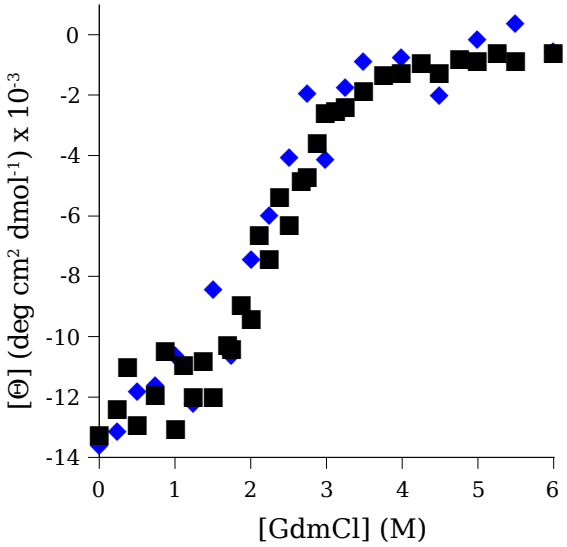

Figure 4.14. Guanidinium chloride induced unfolding of hRXRα(DβE) in the presence of equimolar 9cRA. Monitored using CD spectroscopy at 222 nm. (Black box, ■) 10 μM. (Blue diamond, ◆) 2 μM.

4.3.5 Other Techniques

We also attempted binding assays using isothermal titration calorimetry, but the mixing energy of the organic solvents necessitated by ligand insolubility in water overwhelm the binding signal. Injection of the protein into a solution containing ligand proved irreproducible, possibly due to shearing and/or heating effects when injecting a concentrated, viscous protein solution. Attempts to titrate in the LxxLL peptide were irreproducible, despite the reported success of a similar experiment (295).

Hanging drop crystallization attempts were unsuccessful, even in conditions previously reported to produce crystals (189-192). We attempted to use misincorporation proton-alkyl exchange (MPAX) system and isotope-coded affinity tag (ICAT) mass spectrometry (323) to probe the structural flexibility of hRXRα(DβE), but did not obtain reproducible data.

To further clarify the oligomeric distribution of hRXRα(DβE), gel filtration chromatography and dynamic light scattering were used.

However, the necessary assumptions of size and shape allows for great latitude in interpretation of the data. Although the results are ambiguous, the experiments provided no evidence to contradict the basic analysis from AUC.

4.4 Discussion

4.4.1 Comparison to Prior Work

RXRα is an oligomeric, multidomain protein that exhibits complex behavior. We have characterized some of the ligand binding, self-association, thermal denaturation, coactivator interactions, and thermodynamic properties of hRXRα($D_{\beta}E$). The use of a standard set of conditions allows presentation of a more complete, standardized analysis than has been available to date and can be used to resolve some of the differences between prior reports.

Our measured K_d for 9cra bound to hRXRα($D_{\beta}E$) is higher than most previous measurements (Table 4.2). Several reasons explain why our value is unexpectedly high.

First, the more domains of RXRα used in a binding assay, the lower the measured K_d tends to be (Table 4.2): the K_d range for full-length RXRα is 1.44-13 nM, for RXRα(CDE) 5-41 nM, and for RXRα(DE) and RXRα(E) 1.8-200 nM. Although the LBD is essential for binding the ligand, the other domains apparently have a role in the ligand affinity, which has been suggested by a functional analysis of peroxisome proliferator-activated receptor α (PPARα), where replacement of the PPARα DBD by the yeast GAL4 DBD resulted in a reduced maximum activity but similar overall profile (46). This trend suggests that our K_d value for hRXRα($D_{\beta}E$) is somewhat increased by the absence of the other domains.

Second, many previous reports did not obtain sufficient data points

Some prior work used only 4 or 5 data points to calculate the K_d, and in only a very few cases have the authors used a maximum ligand concentration sufficient to claim "saturation" (a requirement for accuracy in all fitting methods (324)). A simple mathematical analysis of the K_d equation demonstrates that even a ligand concentration 10-fold higher than the K_d will provide a maximum of 90% bound receptor (assuming the receptor concentration is orders of magnitude below the K_d). Insufficient saturation tends to reduce the calculated K_d.

Third, many reports use linearized data during analysis. The use of linearized data violates assumptions required for nonlinear analysis (whether linearized or not) (312, 313); see Chapter 6 for a discussion. The Scatchard (325) and Cogan (326) plots used to analyze all of the radioligand and some of the fluorescence experiments referenced in Table 4.2 are not statistically valid (312, 313, 327). In addition, a linearized analysis requires an extrapolation of several orders of magnitude from the measured data to determine the concentration of bound receptor, which is inherently an undesirable situation and strongly influences the reported K_d value (324). Linearization can have unpredictable effects on the calculated K_d, but tends to enhance the effect of insufficient data (i.e., decrease the estimated K_d).

Fourth, the fluorescence assays suffer from the time-dependent tryptophan quenching shown in Figure 4.3, which was usually interpreted as being ligand-dependent. Failure to correct for this signal loss results in artificially low K_d values. Several investigators have commented on this signal loss (247, 249, 256, 258, 259). One group used cuvettes coated with gelatin to eliminate the loss (249), but no treatment entirely eliminated this loss in our hands. However, it is noteworthy that this report is the only one to claim a K_d (for 9cRA and hRXRα(E)) greater than ours (up to 200 nM). In other cases, the loss has been attributed to "inner filtering" (247, 256, 258,

259), which is signal loss due to significant absorption of the excitation or emission light by the sample. Inner filtering should not be a time-dependent behavior, but rather a constant or concentration-dependent correction for signal loss due to self-absorbance or interference in the buffer, both of which are generally negligible at the concentrations used for the binding assays. Although at low µM concentrations 9cRA does have sufficient absorbance to possibly cause inner filtering, time-dependence alone explains the signal decrease. Some prior reports have used higher concentrations of receptor (approximately 1 µM), and they tend to have higher reported K_d values (247, 258, 259, 262, 264); at µM concentrations, the magnitude of signal is greatly reduced. One explanation for the signal loss is adsorption to the cuvette surface, as attempts to eliminate the signal loss using surface coatings had a greater (often detrimental) effect than changes to the buffer conditions. Adsorption implies a concentration change in the cuvette solution over time, and so analyses based on a fixed receptor concentration become unreliable. Previous attempts to correct for this self-quenching by adjusting the magnitude of the signal (247, 256, 258) address only one of the two variables in the fit that are changing (signal, not protein concentration). This signal loss is probably also a problem in experiments on binding kinetics (257, 263).

We minimized signal loss by minimizing cuvette contact time, used least-squares analysis to avoid linearization, and obtained measurements to a ligand concentration at least 40-fold greater than the K_d were possible. When the maximum obtainable data point was less than approximately 10-fold greater than the K_d, we calculated only a lower limit K_d. Comparisons of analysis methods using different subsets of our data suggest that each of these modifications serve to increase the apparent K_d, with the net result that our K_d is significantly higher than most previously reported values. The

one-sample-per-cuvette protocol has the unfortunate effect of introducing significant noise into the system, which was minimized by obtaining many more measurements than typical. Finally, our error analysis also provides a more realistic and larger value for the error estimates than standard approaches (324).

In addition to the numerous reports examining binding of hRXRα($D_{\beta}E$) to 9cRA, there is also one report of a K_d for LG335 of 269 ± 52 nM with hRXRα(ABCDE), determined via competition with radiolabeled 9cRA (315). This value is within error of our value reported for LG335 (370 ± 60 nM). Our K_d for atRA, 900 ± 300 nM, is also in agreement with prior reports, which indicate the K_d is greater than 200 nM (176, 249, 253, 258, 260). Although atRA has been reported to isomerize in cells to form other isomers, including 9cRA (176), we do not believe isomerization to be cause of our observed binding. Approximately 10% 9cRA contamination would be necessary to generate our results purely from 9cRA, but atRA is more stable than 9cRA (328). Moreover, we detected no significant changes in ligand stocks over the course of several months provided they were kept in the dark and at -20 °C when not in use.

We are not aware of any other reports of determining ligand binding to RXRα via fluorescence quenching using ligands other than 9cRA. One proposed explanation for tryptophan quenching in RXRα is energy transfer between the tryptophan residues and 9cRA (247, 259), which is not unreasonable in light of the overlap in emission and absorption spectra. However, as we have clearly demonstrated, the effect is applicable to many ligands, including one (LG335) with no appreciable absorption overlap with the tryptophan emission spectrum. Therefore, the quenching mechanism is due simply to having a ligand in the binding pocket, possibly due to either tertiary structure rearrangement upon ligand binding and/or direct

quenching via means that do not lead to emission, such as energy transfer to peptide bonds as they shift or the change in charge distribution when the ligand binds (320, 321, 329, 330).

The thermal behavior of nuclear receptors is relatively poorly studied, as only a handful of investigators have discussed thermal denaturation (331-334). Where reported, all observed irreversible denaturation behavior similar to the denaturation of hRXRα($D_{\beta}E$). hRXRα($D_{\beta}E$) is apparently thermally more stable (55 °C) than the estrogen receptors (38-40 °C) (331, 334) and PPARγ (46 °C) (333), although not necessarily retinoid-related orphan receptor α (61 °C with low-affinity ligand) (332). A few investigators have also reported a ligand-dependent increase in the T_m of receptors (331-333). ΔT_m values range from < 1 °C to 7 °C for PPARγ (333), up to at least 9 °C for retinoid-related orphan receptor α (332), and up to 5 °C for estrogen receptor α (331), which are comparable to our maximum of nearly 7 °C, with 9cRA (Table 4.6).

The oligomerization of RXRα has been examined in more detail than ligand binding (Table 4.1). We are aware of only two investigations of RXRα by AUC; one agrees well with our results despite the use of a slightly different construct and different assay conditions (265), while the other suggested a predominantly monomeric species (249), in disagreement with nearly all later publications. Only a few quantitative reports for the dimer-tetramer interaction exist, although they are quite different from ours (20 ± 2 μM): for hRXRα(CDE), an approximate value of a few hundred micromolar has been reported (188), while for mRXRα(CDE) values of 4.4 nM and 2.8 nM were found using fluorescence anisotropy (21, 257). The lack of the DBD in our construct may explain our much lower affinity, as the DBD alone is known to oligomerize (7). The K_d for the monomer-dimer equilibrium has also been reported for mRXRα(CDE): 155 nM (21), 130 nM (258), and 0.38

nM (269) in the absence of ligand, and 63 nM in the presence of 9cRA (258). The prior monomer-dimer results agree with our upper limit of 200 nM. Other differences between our results and prior work may be caused by the uniformity of our hRXRα($D_\beta E$) following expression, which does not show non-equilibrium effects that have previously been reported (192, 201, 256).

The chemical denaturation data presented in Figures 4.13 and 4.14 appears to be in agreement with the AUC oligomerization results, despite the lack of a quantitative analysis. The difference in the baseline CD signal between 2 and 10 µM in the absence of 9cRA is suggestive of a decrease in the tetramer concentration in favor of the dimer, as would be expected; the lack of such a difference in the presence of 9cRA suggests an unchanged state, presumably a pure dimer (21, 192, 201, 244, 248, 261, 262, 265, 266, 269, 271, 272, 293, 294).

Prior work examining chemical denaturation of hRXRα($D_\beta E$) used a model lacking a tetrameric species (269). Whether by good planning or good fortune, our K_d for the dimer-tetramer reaction may justify their choice of model, as their concentration used during unfolding was low enough that at most a few percent of the receptor would have been tetrameric. The use of an elevated temperature (30 °C) may have further reduced the tetramer component. Our data do not justify eliminating the tetramer species from the model, but even a model excluding it does not provide a reliable fit to the data.

4.4.2 Correlations

Previous work has suggested a correlation between the affinity of a ligand and ΔT_m of the receptor (331-333). In the one case where a direct linear correlation has been drawn, using PPARγ, an R^2 of 0.95 was found (333). As shown in Figure 4.15A, there is a definite trend of increased ΔT_m as affinity increases, but it is not a statistically significant correlation (R^2 =

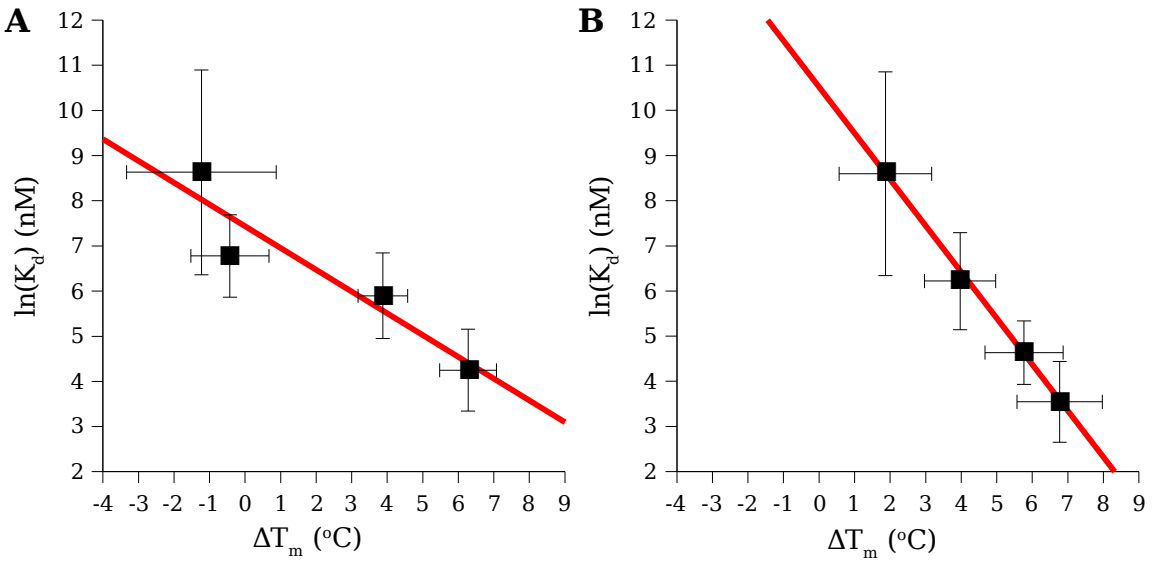

Figure 4.15. Linear correlations of hRXRα(DβE) thermal stability and ligand affinity. (A) Measurements in the absence of LxxLL peptide. (B) Measurements in the presence of LxxLL peptide. (Red lines, ▬) Linear fits.

0.876, p = 0.07). For the data measured in the presence of the LxxLL peptide, the correlation is significantly better (Figure 4.15B, R^2 = 0.997, p = 0.0014). However, unlike previous work that only examined agonists, we find that this trend is true regardless of whether or not the ligand activates the receptor. Only the affinity for the receptor seems to be important in determining the magnitude of ΔT_m. The ΔT_m may serve as a reasonable proxy for affinity in the absence of a direct assay, but not necessarily activity. One caveat on this analysis is that the correlation uses only four data points, and it is not certain the trends would remain as strong in a larger data set.

Although a good correlation exists between the EC_{50} values (the concentration of ligand at 50% of the maximum activity) for and the K_d values for 9cRA, LG335, and OPBA, it is not statistically significant; atRA must be excluded because it does not activate RXRα. The magnitude of the effect of the LxxLL peptide on the K_d values does not correspond to how effective the ligands are. The inability to explain receptor activity by

binding affinity could be due to our small sample size or the need to include more complex interactions (e.g., oligomerization).

4.4.3 Implications

We have presented an improved protocol for obtaining accurate K_d values for RXRα. In addition, we have shown that fluorescence quenching is a general property of ligand binding to RXRα, and therefore can be used in place of more expensive assays. Our results and review of prior work suggest that while full-length RXRα may have somewhat greater affinity for 9cRA than hRXRα($D_\beta E$), in general prior K_d estimates have been significantly too low. The observation that the LxxLL peptide reduces the K_d may partly explain why assays done from mammalian cell lysates tend to have a lower reported K_d, as coactivator proteins will also be present in the lysate.

The chemical denaturation data suggests the monomer of hRXRα($D_\beta E$) is not highly stable, as approximately 3.5 M GdmCl is sufficient to yield the final unfolded state as measured by CD or fluorescence spectroscopy. Addition of 9cRA may eliminate some of the transitions (oligomeric states) during the unfolding process, but does not seem to cause an overall increase in thermodynamic stability. Although the final transition in Figure 4.13 (between 2.5 and 3.5 M GdmCl) may correspond to the monomer-unfolded equilibrium, it is also possible that the folded monomer does not exist in significant concentration at any time during the unfolding process. The relatively shallow transition slopes (the transitions take place over a nearly 1 M GdmCl concentration range) suggests that the transitions are not highly cooperative.

The correlation between ΔT_m and K_d might be interpreted in light of a simple model whereby hRXRα($D_\beta E$) becomes more rigid upon ligand binding, and therefore less likely to denature. Indeed, a reduction in flexibility upon ligand binding has been proposed as a general model for nuclear receptors

(41). However, several lines of argument suggest something more complex may be occurring First, some reports have examined the flexibility of hRXRα(E) in the presence of various ligands using hydrogen-deuterium exchange mass spectrometry (268, 272, 273, 275); as a whole, these reports have not found a strong correlation between ligand affinity and either the amount of exchange protection (i.e., reduction in flexibility) or the particular sites protected (near or remote from the binding pocket). 9cRA is still the highest affinity ligand known for RXRα, yet it does not cause the greatest degree of protection (275). Second, similar results are observed crystallographically, where a comparison of hRXRα(E) bound to 9cRA and other ligands shows that lower affinity ligands actually have more receptor contacts than 9cRA (192). Third, in-solution results suggest that the receptor retains a great deal of flexibility even when bound to ligand: although proteolysis experiments show increased resistance to proteases when mRXRα(CDE) is bound to 9cRA (248), NMR structures of hRXRα(E) in the presence and absence of 9cRA both show a great deal of flexibility (201). The apparent T_m is a value under the control of kinetic processes, so we suggest that instead of a global "more rigid" model, it is probable that some key process in the aggregation (which may or may not involve "unfolding") has a significantly reduced rate when ligand is bound. This idea is further supported by the apparent lack of a thermodynamic stabilization of the receptor by 9cRA.

We have also shown unambiguously that hRXRα($D_\beta E$) exists, at least at micromolar concentrations, in a dimer-tetramer equilibrium. Although the 20 μM dissociation constant is far higher than what might seem necessary to support claims of a physiological role for RXRα tetramers (262, 271), we can make no observations regarding a (likely) reduction in the K_d when the additional domains in RXRα are present and interacting, or of

possible *in vivo* effects due to cellular crowding that may also tend to reduce the K_d.

CHAPTER 5
CHARACTERIZATION OF hRXRα(DβE) VARIANTS

5.1 Introduction

Although much effort has gone into finding new ligands for RXRα and understanding the effect of those ligands on the receptor's function, relatively little work has been performed from the alternate viewpoint: how does changing the receptor affect its function, self-association, and interactions with ligands? This perspective allows us to ask more fundamental questions about the protein, such as which residues are critical in discriminating between ligands. Phylogenetic analysis can provide somewhat similar information; the most ambitious effort to date has been performed using the mineralocorticoid and glucocorticoid receptors, where an extensive analysis suggests that the two receptors evolved from a single ancestor protein with the ability to bind both mineralocorticoid and glucocorticoid receptor ligands, but with mineralocorticoid receptor-like function (335). On a smaller scale, a few studies have examined the biophysical effects of specific, naturally occurring mutations in RXRα (216, 262, 265).

The Doyle laboratory has engineered a large number of hRXRα(E) variants, using a variety of techniques, with the goal of creating a variant protein that responds to a new ligand (216-221). While many receptors have been found that are activated by the synthetic ligand LG335, and a few that are activated by OPBA, it is not immediately obvious why the mutations should cause the observed changes in ligand-dependent activation. The simplest explanation is that the binding affinity of the variant receptors for the target ligand is higher than the wild-type affinity, but the effects of the mutations are not necessarily limited to ligand affinity. Changes in DNA

Table 5.1. Mutations in hRXRα(DβE) variants.

	Mutated Amino Acids						
Variant	**I268**	**A272**	**Q275**	**I310**	**F313**	**L436**	**F439**
AMAT	A			M	A	T	
ASAF	A			S	A	F	
VVLM	V	V		L	M		
VVMSM	V	V		M	S	M	
SVM		S		V	M		
TLT		T		L	T		
CMI			C	M	I		
LMM				L	M	M	
IL					I		L

binding affinity, oligomeric distribution, and coactivator recruitment are among the other possible effects. Examples of these alternative effects have been shown most clearly for the androgen receptor. Specific mutations in the androgen receptor have been shown to change the binding kinetics but not the overall K_d (336), significantly reduce the *in vivo* activity of the protein at elevated temperatures (337), and cause changes in coactivator interactions only for a specific ligand (338).

Therefore, we performed an extensive biophysical characterization on nine functional variants of hRXRα(DβE) (see Table 5.1) to try to understand what may be driving the observed changes in activity. The variants, which have a wide range of EC_{50} and efficacy values (see Table 5.2), were also engineered using a variety of methods, including site-directed mutagenesis and genetic selection from two large libraries of engineered receptors. The variants chosen have mutations throughout the binding pocket of hRXRα(DβE) (see Figure 5.1).

Table 5.2. Activity data for hRXRα(DβE) variants in yeast.

		9cRA		LG335		OPBA	
Variant	**Ref.[β]**	**EC_{50} (nM)**	**Eff.[α] (%)**	**EC_{50} (nM)**	**Eff.[α] (%)**	**EC_{50} (nM)**	**Eff.[α] (%)**
WT	(219)	100	100	300	10	70,000	20
AMAT	(219)	> 10,000	0	610	10	N.D.[δ]	N.D.[δ]
ASAF	(219)	> 10,000	0	430	50	N.D.[δ]	N.D.[δ]
VVLM	(219)	> 10,000	10	40	60	N.D.[δ]	N.D.[δ]
VVMSM	(219)	> 10,000	10	680	30	N.D.[δ]	N.D.[δ]
SVM	(221)	> 10,000	10	> 10,000	0	2000	100
TLT	(221)	> 10,000	0	> 10,000	0	3000	80
CMI	(220)	> 10,000	10	1000	100	N.D.[δ]	N.D.[δ]
LMM	(221)	600	90	20	190	N.D.[δ]	N.D.[δ]
IL	(220)	N/A[γ]	N/A[γ]	N/A[γ]	N/A[γ]	N.D.[δ]	N.D.[δ]

[α] Eff. = Efficacy, relative to wild-type hRXRα and 9cRA.
[β] Ref. = Reference.
[γ] N/A = Not applicable: receptor is active without ligand (although it is further activated by ligands).
[δ] N.D. = No data.

5.2 Materials and Methods

5.2.1 Materials

Materials were used as described in Section 4.2.1. Proteins were expressed and purified as described in Section 3.2.3. The mutations in each variant are shown in Table 5.1. Yields were between 40 and 80 mg L^{-1} with > 95% purity for all variants except AMAT, which gave yields of only 10-15 mg L^{-1} with purities of approximately 90%.

5.2.2 Ligand Binding

Binding assays were done as described in Section 4.2.2. LG335 and 9cRA binding constants were measured for all variants, while OPBA was measured only with the variants it is known to activate, SVM and TLT. Ligands with an affinity too low to reliably measured are given as a lower

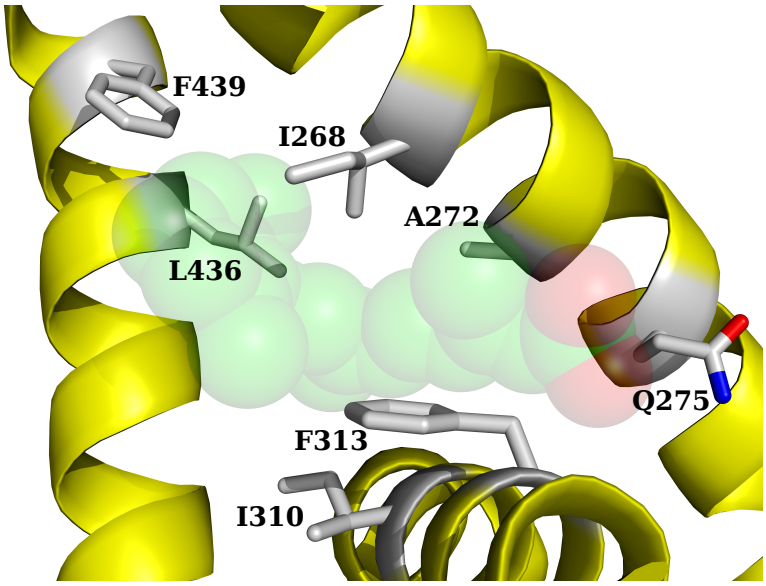

Figure 5.1. Residues mutated in at least one hRXRα($D_{\beta}E$) variant. 9cRA is represented as the semi-transparent spheres. Wild-type amino acids are shown. For clarity, only helices 3, 5, and 11 are visible. As shown, A272, Q275, and L436 are partially or completely behind the ligand. Image constructed from Protein Databank entry 1FBY (190).

limit of the K_d, as described in Section 4.3.1. For a few ligand-receptor pairs, the presentation of symmetric errors for K_d values implied inclusion of "0 nM" as statistically possible, and so the asymmetric errors are shown for these values. See Sections 6.2 and 6.3.1 for details.

5.2.3 Thermal Denaturation

Thermal denaturation was monitored by CD spectroscopy and DSC as described in Section 4.2.4. The effect of OPBA on the T_m was only measured for variants known to be activated by it.

5.2.4 Self-Association

Equilibrium AUC was performed as described in Section 4.2.3. The partial specific densities and molecular masses of the variant are shown in Table 5.3. The limits for monomer-dimer and dimer-tetramer equilibria were extracted from the monomer-tetramer fit for a few variants using Equation

Table 5.3. Physical data for hRXRα($D_{\beta}E$) variants.

Variant	$\bar{\nu}$ [α] (ml mg^{-1})	Molecular Mass (Da)
AMAT	0.7367	29,034
ASAF	0.7366	29,036
VVLM	0.7389	29,145
VVMSM	0.7374	29,136
SVM	0.7382	29,132
TLT	0.7384	29,130
CMI	0.7387	29,115
LMM	0.7382	29,148
IL	0.7398	29,078

[α] Partial specific volume.

5.1:

$$K_{d1,4}=K_{d1,2}^{2}K_{d2,4}=n^{2}K_{d2,4}^{3} \tag{5.1}$$

where $K_{d1,4}$ is the monomer-tetramer dissociation constant; $K_{d1,2}$ is the monomer-dimer dissociation constant; $K_{d2,4}$ is the dimer-tetramer dissociation constant; and n is the minimum ratio of $K_{d1,2}$ to $K_{d2,4}$. For these calculations, the error estimates were applied in the most conservation fashion, so the $K_{d2,4}$ threshold was calculated assuming 1 SD positive error, while $K_{d1,2}$ was calculated assuming 1 SD negative error, with the result that the actual difference in the thresholds is less than n. We used an n of 100, which implies a dimer concentration below the level of reliable detection (5%).

5.2.5 Chemical Denaturation

Chemical denaturation was monitored by CD spectroscopy, as described in Section 4.2.5.

5.2.6 Data Analysis

Nonlinear and linear data analysis was performed as described in Section 4.2.6.

5.2.7 Prediction of Key Residues

Our approach to identifying potentially key residues was modeled after prior work (339-341). In short, we applied a nonlinear least squares analysis to Equation 5.2:

$$\Delta\Delta G = \sum_{i=1}^{n} X_i c_i \tag{5.2}$$

where n is the total number of mutations in all variants; X_i is either 0 or 1 to indicate the absence or presence, respectively, of mutation i in the current variant; and c_i is the numerical constant representing the effect of mutation i. The experimental ΔΔG for each variant was calculated according to Equation 5.3:

$$\Delta\Delta G = RT \ln\left(\frac{\Theta_{var}}{\Theta_{wt}}\right) \tag{5.3}$$

where R is the universal gas constant; T is the temperature in Kelvin; Θ_{var} is the measured value for the variant; and Θ_{wt} is the measured value for wild-type hRXRα(DβE). Efficacy and EC_{50} data were used for Θ as described in Section 5.3.5, and T was set to 303 K, as the yeast cells were grown at 30 °C. As discussed in Section 5.3.5, there were more variables (mutations) than data points (sequenced variants), so nonlinear least squares produced infinite solutions of equal statistical validity. To produce a single solution, we fit the data 50 times starting from random initial guesses and the 10 best solutions, as evaluated by χ^2 (to eliminate solutions trapped in a non-equivalent local minima), were kept. An estimate for each c_i was calculated by taking the mean and standard deviation of the ten solutions. A particular estimate was considered robust if it met the following criteria: (i) the

elimination of various combinations of two or more variants from the fit data did not lead to a significantly different prediction; (ii) the sign of the parameter was the same for all ten best fit solutions; and (iii) the estimated value was at least two standard deviations from zero.

The variants were constructed using standard QuikChange® mutagenesis protocols, and the function evaluated in yeast as described elsewhere (219, 220).

5.3 Results

5.3.1 Ligand Binding

Fluorescence quenching data to obtain ligand binding constants for each receptor-ligand pair are shown in Figure 5.2. The K_d values for each pair are shown in Table 5.4, as are wild-type values for comparison. All variants had reduced affinity for 9cRA compared to wild-type; the largest reduction is at least 40-fold. Four variants, AMAT, VVLM, SVM, and LMM, had significantly greater affinity for LG335 than wild-type hRXRα($D_\beta E$), while two variants, ASAF and CMI, had significantly reduced affinity. Neither variant activated by OPBA had high enough affinity for the K_d to be measured confidently; the lower boundaries on the K_d are similar to wild-type hRXRα($D_\beta E$).

All variants were fit to a single-binding model assuming one binding site per monomer independent of the actual oligomeric state, but a few variant-ligand pairs do not seem to be adequately fit by this model. The most extreme example is LMM, especially with LG335, where the shape of the data are suggestive of two binding events and the fit line is apparently missing the higher concentration data completely (see Figures 5.2 and 5.3). However, only a small number of variant-ligand pairs have significantly improved fits with alternative models, the consistency of results when those

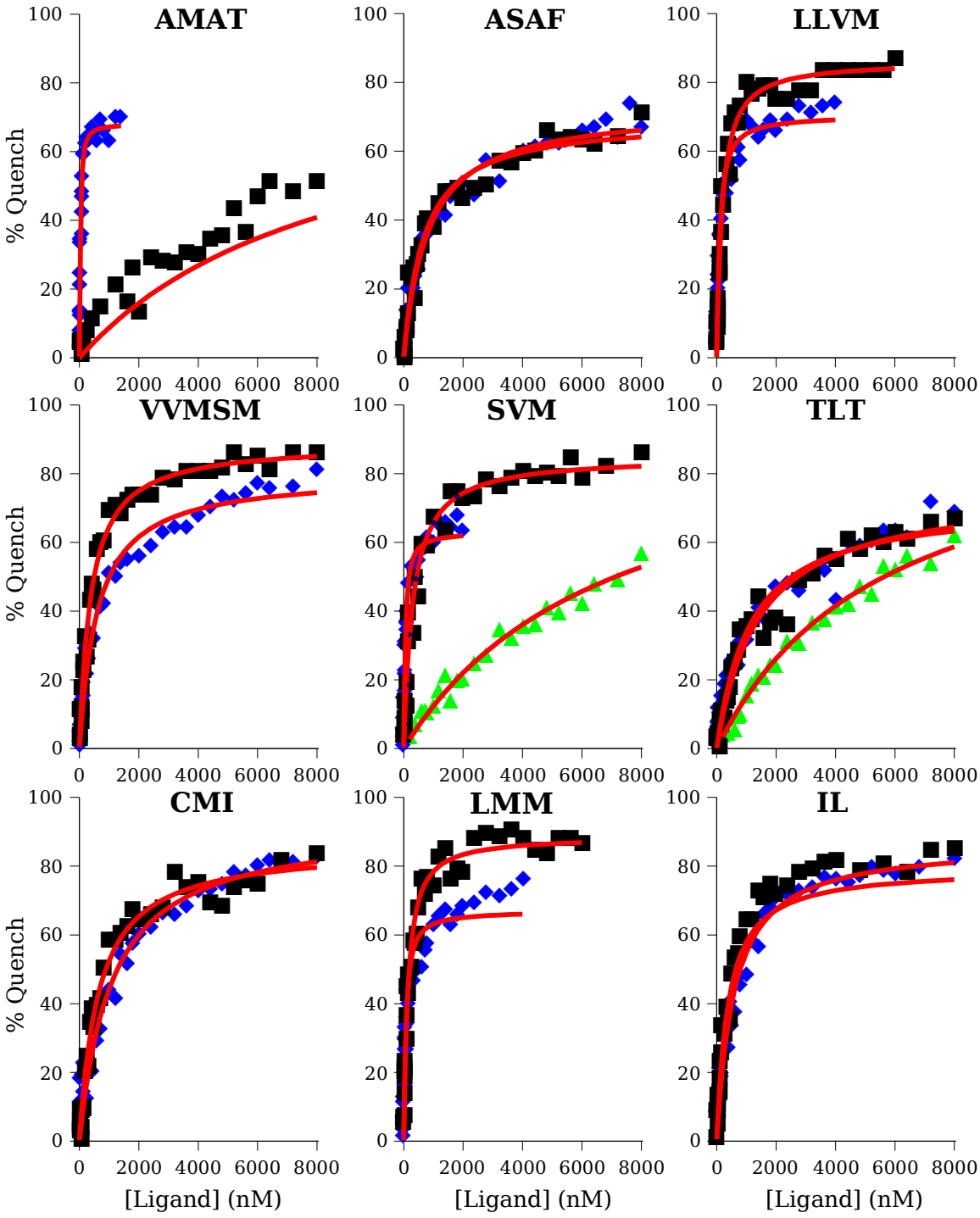

Figure 5.2. Ligand binding data and fits for 50 nM hRXRα($D_{\beta}E$) variants. (Black box, ■) 9cRA. (Blue diamond, ◆) LG335. (Green triangle, ▲) OPBA. (Red lines, ▬) Best fit curves. The upper limit of data collection is 8000 nM or at least 40*K_d.

Table 5.4. Ligand binding parameters for hRXRα($D_{\beta}E$) variants.

	9cRA		LG335		OPBA	
Variant	**K_d (nM)**	**Max Quench (%)**	**K_d (nM)**	**Max Quench (%)**	**K_d (nM)**	**Max Quench (%)**
WT[α]	70 ± 20	84 ± 4	370 ± 100	83 ± 5	> 3800	65
AMAT	> 2800[β]	60	16 ± 7[γ]	68 ± 3		
ASAF	600 ± 200[β]	69 ± 6	800 ± 300[β]	73 ± 6		
VVLM	160 ± 40[β]	86 ± 4	80 ± 40[γ]	71 ± 5		
VVMSM	360 ± 70[β]	89 ± 4	580 ± 170	80 ± 5		
SVM	280 ± 80[β]	85 ± 4	40 ± 30[γ]	63 ± 5	> 3700	70
TLT	1400 ± 700[β]	76 ± 10	1100 ± 700	72 ± 12	> 2900	70
CMI	610 ± 150[β]	86 ± 5	1100 ± 400[β]	92 ± 8		
LMM	130 ± 30[β]	89 ± 3	60 (+60, -25)[γ]	67 ± 7		
IL	360 ± 80[β]	87 ± 4	600 ± 400	75 ± 8		

[α] For comparison purposes; see Section 4.3.1.
[β] Significantly lower affinity than wild-type.
[γ] Significantly greater affinity than wild-type.

models were applied is poor, and no strong statistical justification exists for alternative models (as described in Section 4.3.1). Therefore, we continued to apply only the simplest model. Without additional information, it is not possible to determine whether the deviations in the data are due to an actual difference in the binding mode, or instead reflect a more complex tertiary structure change upon ligand binding that results in deviations from the expected quenching behavior.

The affinity for all ligand-receptor pairs was also measured in the presence of a fixed concentration (20 μM) of LxxLL peptide; data are presented in Figure 5.3 and Table 5.5. Unlike the wild-type hRXRα($D_{\beta}E$), most ligand-variant pairs did not have a significant decrease in the K_d in the presence of the LxxLL peptide: only ASAF, TLT, and IL had increased affinity for LG335, and CMI and IL for 9cRA. Although the lower limit of the K_d for TLT and SVM with OPBA is decreased, it is not possible to conclusively state

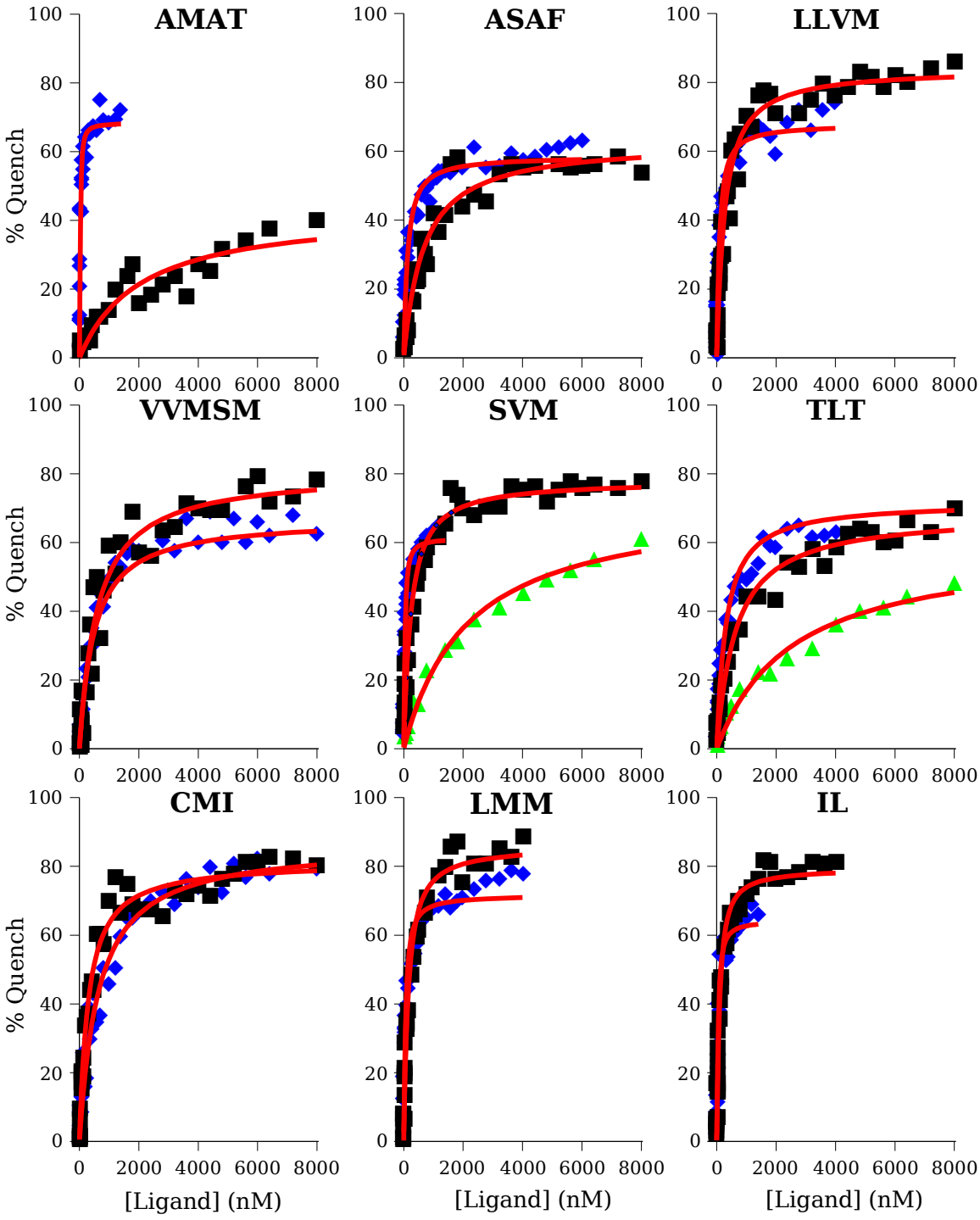

Figure 5.3. Binding data and fits for 50 nM hRXRα(D$_{\beta}$E) variants in the presence of 20 μM LxxLL peptide. (Black box, ■) 9cRA. (Blue diamond, ◆) LG335. (Green triangle, ▲) OPBA. (Red lines, ▬) Best fit curves. The upper limit of data collection is 8000 nM or at least 40*K_d.

Table 5.5. Ligand binding parameters for hRXRα($D_{\beta}E$) variants in the presence of 20 μM LxxLL peptide.

	9cRA		LG335		OPBA	
Variant	**K_d (nM)**	**Max Quench (%)**	**K_d (nM)**	**Max Quench (%)**	**K_d (nM)**	**Max Quench (%)**
WT[α]	35 ± 14[β]	75 ± 4	110 ± 30[β]	71 ± 3	> 3800	55
AMAT	> 3500	55	9 (+10, -4.4)	69 ± 5	N.D.[γ]	N.D.[γ]
ASAF	600 ± 300	63 ± 6	110 ± 50[β]	59 ± 3	N.D.[γ]	N.D.[γ]
VVLM	240 ± 90	84 ± 5	70 ± 50	68 ± 6	N.D.[γ]	N.D.[γ]
VVMSM	600 ± 300	81 ± 8	460 ± 120	70 ± 4	N.D.[γ]	N.D.[γ]
SVM	210 ± 80	78 ± 4	20 ± 14	62 ± 4	> 1900	70
TLT	600 ± 200	69 ± 5	280 ± 110[β]	72 ± 5	> 2000	55
CMI	270 ± 80[β]	81 ± 4	600 ± 200	87 ± 7	N.D.[γ]	N.D.[γ]
LMM	130 ± 60	86 ± 5	40 (+40, -16)	72 ± 6	N.D.[γ]	N.D.[γ]
IL	70 ± 20[β]	80 ± 4	26 ± 19[β]	64 ± 6	N.D.[γ]	N.D.[γ]

[α] For comparison purposes; see Section 4.3.1.
[β] Significantly greater affinity than in the absence of LxxLL peptide.
[γ] N.D. = No data.

that the K_d has decreased. As was observed for wild-type hRXRα($D_{\beta}E$), variant-ligand pairs tested with a higher concentration of LxxLL peptide did not have significant differences from measurements made with 20 μM LxxLL. Figure 5.4 shows the shift in K_d values due to the LxxLL peptide graphically, where ligand-receptor pairs with significantly changed affinity (due to the peptide) are below the red line.

5.3.2 Thermal Stability

All variants irreversibly denatured into apparently similar, β-sheet-like aggregates upon heating. The variants span a 20 °C range of apparent T_m values (Table 5.6). Most variants were more stable (higher apparent T_m) than wild-type, although ASAF was significantly less stable.

Table 5.6 also shows the ΔT_m for each variant-ligand pair. All variants

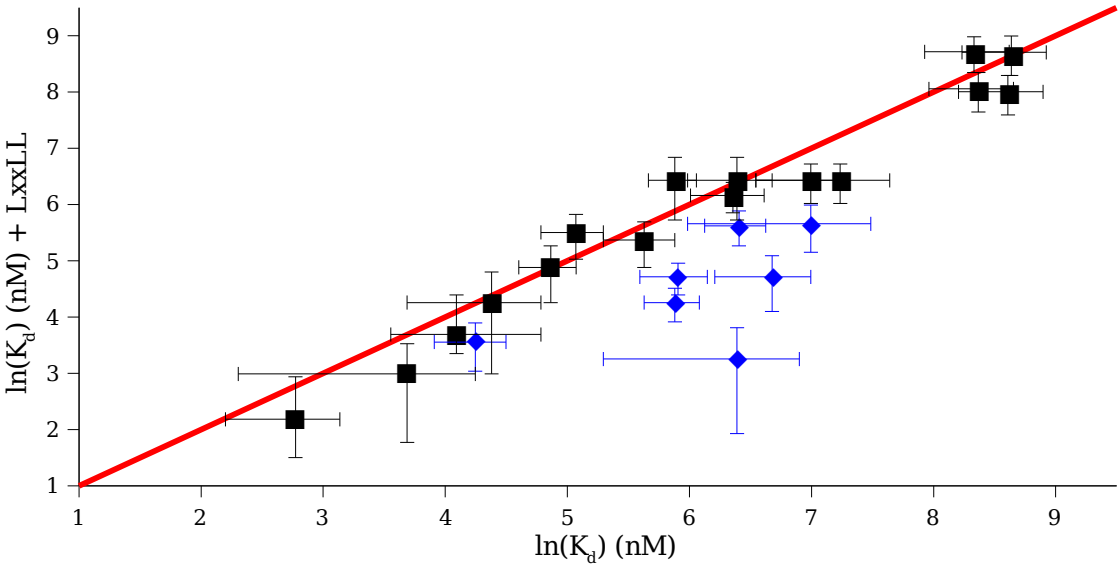

Figure 5.4. Effect of the LxxLL peptide on ligand affinity for hRXRα(DβE) variants. (Red line, ▬) Visual indicator of region of no significant change. (Black boxes, ■) Ligand-receptor pairs that do not have significant differences in assays with and without LxxLL peptide. (Blue diamonds, ◆) Ligand-receptor pairs that have significantly higher affinity when the LxxLL peptide is present.

were less stabilized by 9cRA than was wild-type hRXRα(DβE). AMAT was apparently destabilized by 9cRA, although the immediate cause for this is not clear, as experiments performed by neither CD spectroscopy nor DSC gave any clear indication of ligand binding to the denatured state. OPBA did not stabilize either SVM or TLT, but, similarly to wild-type hRXRα(DβE), showed evidence of the ligand binding to the aggregate state. Only one variant, TLT, was less stabilized by LG335 than wild-type hRXRα(DβE). AMAT, ASAF, and VVLM were all significantly more stabilized by LG335 than wild-type hRXRα(DβE).

The effect of the LxxLL peptide on variant T_m values was quite diverse (see Table 5.7). Only one variant, VVLM, was significantly stabilized by the LxxLL peptide alone. As with wild-type hRXRα(DβE), the certainty in the apparent T_m is relatively poor. ASAF and IL had an additional increase, beyond that of the ligand alone, in the 9cRA ΔT_m when the LxxLL peptide

Table 5.6. Thermal stabilities and effects of ligands for hRXRα($D_{\beta}E$) variants.

		+9cRA	+LG335	+OPBA
Variant	**T_m (°C)**	**ΔT_m (°C)[α]**	**ΔT_m (°C)[α]**	**ΔT_m (°C)[α]**
WT[β]	55.0 ± 0.7	6.3 ± 0.8[ε]	3.9 ± 0.7[ε]	-1.2 ± 2.1
AMAT	55.8 ± 0.6	-1.1 ± 0.6	6.8 ± 0.7[ε]	N.D.[ζ]
ASAF	45.7 ± 0.6[γ]	-0.4 ± 0.9	5.4 ± 0.7[ε]	N.D.[ζ]
VVLM	64.2 ± 1.0[δ]	2.2 ± 0.9[ε]	7.3 ± 1.0[ε]	N.D.[ζ]
VVMSM	54.3 ± 0.3	1.9 ± 1.1	2.4 ± 0.5[ε]	N.D.[ζ]
SVM	56.4 ± 0.8	1.6 ± 1.0	4.7 ± 1.2[ε]	0.7 ± 1.1
TLT	58.8 ± 0.6[δ]	-0.3 ± 1.3	0.5 ± 0.9	0.0 ± 1.3
CMI	62.0 ± 1.8[δ]	2.6 ± 2.9	2.1 ± 2.0	N.D.[ζ]
LMM	66.0 ± 1.9[δ]	1.9 ± 1.4	5.5 ± 1.4[ε]	N.D.[ζ]
IL	59.9 ± 0.8[δ]	1.6 ± 1.4	5.3 ± 1.4[ε]	N.D.[ζ]

[α] ΔT_m indicates the change in T_m due to the presence of ligand.
[β] For comparison purposes; see Section 4.3.3.
[γ] Stastically significantly lower T_m than wild-type.
[δ] Stastically significantly greater T_m than wild-type.
[ε] Stastically significant increase in T_m due to ligand.
[ζ] N.D. = No data.

was added. AMAT had a significant decrease. The ΔT_m with LG335 significantly decreased for two variants, AMAT and VVLM, in the presence of the LxxLL peptide; unlike wild-type hRXRα($D_{\beta}E$), none had an increase.

5.3.3 Self-Association

The nine variants were each examined by AUC at three concentrations and three speeds. Like the wild-type protein, most of the variants showed evidence of significant non-specific aggregation at speeds greater than 12,000 rpm. For consistency, the results presented here use only the 12,000 rpm, although adding the 16,000 and 20,000 rpm data to the fits does not change the qualitative results. Raw data and best fits are shown in Figure 5.5. Table 5.8 lists the specific models that were chosen for the variants, and the resulting fit parameters.

Table 5.7. Thermal stabilities and effects of ligands for hRXRα($D_{\beta}E$) variants in the presence of 20 μM LxxLL peptide.

		+9cRA	+LG335	+OPBA
Variant	**ΔT_m (°C)[α]**	**ΔT_m (°C)[α]**	**ΔT_m (°C)[α]**	**ΔT_m (°C)[α]**
WT[β]	3.3 ± 2.3	6.8 ± 1.2[δ]	5.8 ± 1.1[δ]	1.9 ± 1.3
AMAT	0.0 ± 0.8	-4.6 ± 1.0[ε]	4.5 ± 0.8[δ]	N.D.[ζ]
ASAF	2.9 ± 2.1	3.8 ± 1.0[δ]	4.7 ± 0.7[δ]	N.D.[ζ]
VVLM	2.7 ± 1.2[γ]	0.6 ± 1.0	4.7 ± 1.1[δ]	N.D.[ζ]
VVMSM	0.5 ± 1.5	4.0 ± 0.8[δ]	2.6 ± 0.6[δ]	N.D.[ζ]
SVM	3.1 ± 1.8	1.6 ± 0.7[δ]	4.1 ± 0.7[δ]	1.4 ± 1.0
TLT	0.8 ± 0.7	0.1 ± 1.1	1.1 ± 1.0	0.9 ± 1.3
CMI	2.7 ± 2.9	2.2 ± 2.6	-0.5 ± 1.9	N.D.[ζ]
LMM	-0.2 ± 2.7	1.5 ± 1.2	3.5 ± 1.6[δ]	N.D.[ζ]
IL	-0.1 ± 1.7	4.4 ± 1.1[δ]	5.1 ± 1.2[δ]	N.D.[ζ]

[α] ΔT_m indicates the change in T_m due to the presence of lxxll peptide and ligands.
[β] For comparison purposes; see Section 4.3.3.
[γ] Stastically significant increase in Tm due to LxxLL peptide.
[δ] Stastically significant increase in T_m due to ligand and LxxLL peptide.
[ε] Stastically significant decrease in T_m due to ligand and LxxLL peptide.
[ζ] N.D. = No data.

Variants were fit to a variety of a models, and the simplest (fewest species) model giving a reasonable fit was chosen. We defined reasonable to mean that the quality of the fit was not statistically improved by inclusion of additional species in the model. The results from these fits are consistent with qualitative results observed by native PAGE. For the variants best fit by the dimer-tetramer model, the lower limit for reliable measurement of the monomer-dimer K_d was approximately 0.2 μM, given the noise in the measurements and concentrations used. 0.2 μM is therefore an upper limit on the K_d for these variants (SVM and IL), as well as both the dimer-tetramer and monomer-dimer constants for VVLM. The most likely explanation for the variants best fit by a monomer-tetramer model is that the dimer forms the tetramer more readily than the monomer forms the dimer, and therefore

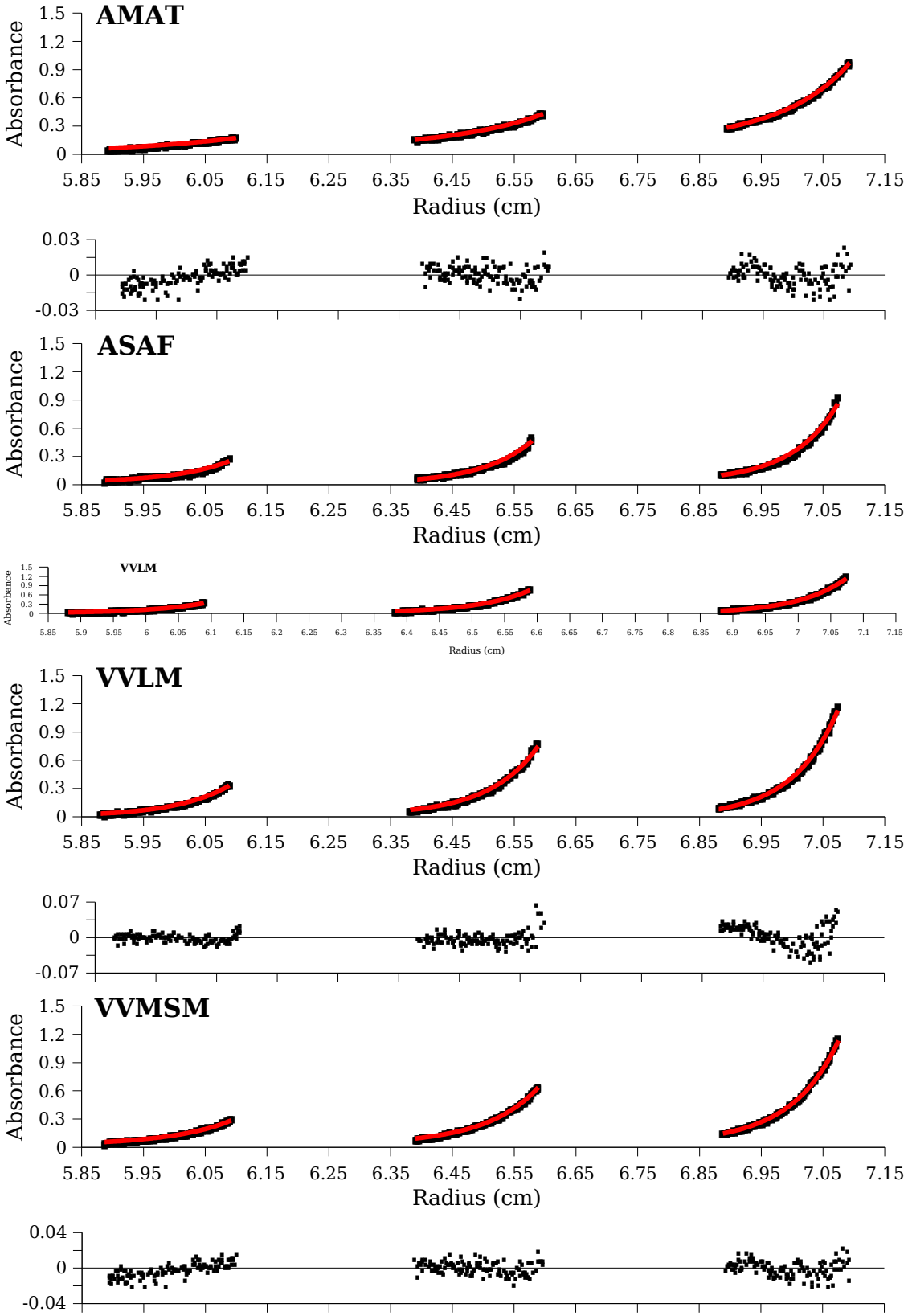

Figure 5.5. AUC data for hRXRα(DβE) variants and fits. (Red lines, ▬) Best fits. See text for specific models.

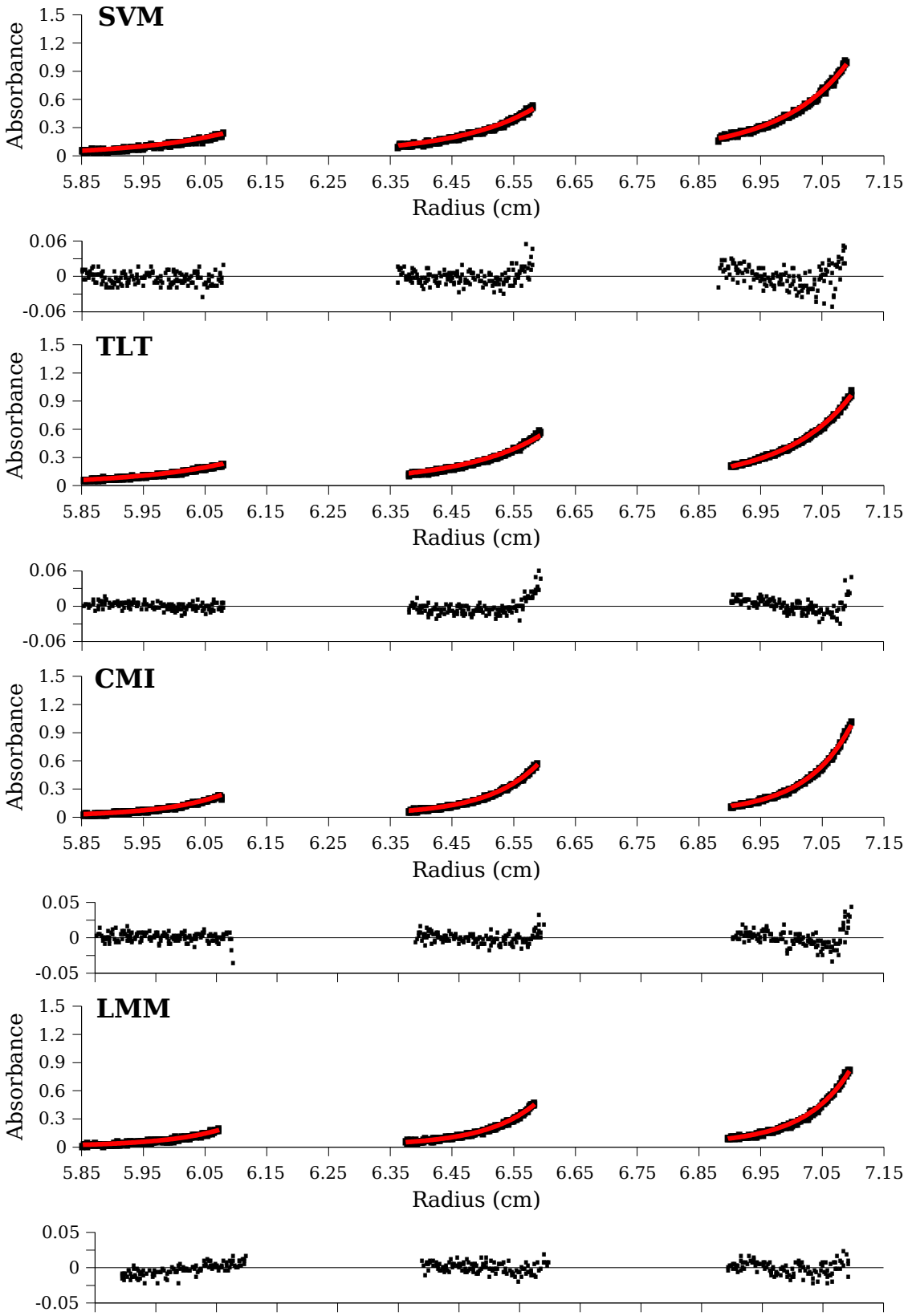

Figure 5.5 (continued)

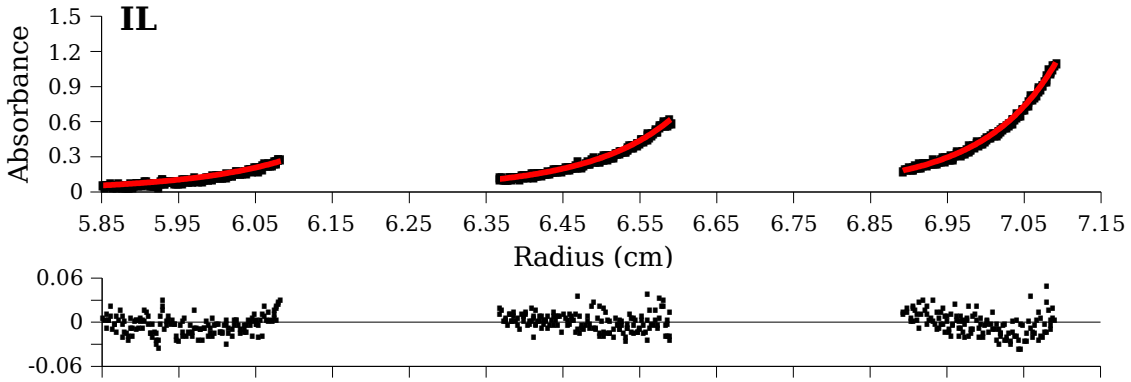

Figure 5.5 (continued)

the dimer is present only in very low concentrations (see Section 5.2.4 for details). The relatively large uncertainty in the best-fit K_d values for CMI and LMM arises because the monomer-tetramer K_d includes the uncertainties for both the monomer-dimer and dimer-tetramer transitions.

The ASAF variant is best fit by a monomer-tetramer model, but the residual patterns and large error estimates for the variant indicate that even this model is not entirely adequate. The poor fit (of any model) is probably due to the formation of higher-order aggregates, which are apparent in native PAGE experiments (data not shown) and suggested by the residual patterns.

The AMAT variant was the only variant to have an average molecular mass below that of a dimeric species, due primarily to a greatly reduced monomer-dimer affinity. The much higher fraction of monomer for this variant than the others may explain why it is isolated with much lower yield and purity. Monitoring fractions taken during purification of AMAT suggest a lower affinity for the metal resin, as significant AMAT was apparent in all the wash fractions (data not shown).

5.3.4 Chemical Denaturation

Variants were denatured using GdmCl and the unfolding monitored by

Table 5.8. Global fit parameters of AUC data for hRXRα(DβE) variants.

Variant	Average Molecular Mass (kDa)	Best Model[α]	K_d[α]		
			T ⇌ 4M (μM^3)	T ⇌ 2D (μM)	2D ⇌ 4M (μM)
WT[β]	80.0 ± 1.1	2D ⇌ T		20 ± 2	< 0.2
AMAT	52.2 ± 1.8	4M ⇌ 2D ⇌ T		13 ± 2[γ]	40.7 ± 0.4[δ]
ASAF	103 ± 4	4M ⇌ T	13 ± 12	< 0.14[γ]	> 5[δ]
VVLM	113 ± 3	T		< 0.2[γ]	< 0.2
VVMSM	94.6 ± 1.8	4M ⇌ 2D ⇌ T		1.39 ± 0.01[γ]	1.32 ± 0.01[δ]
SVM	67.6 ± 1.8	2D ⇌ T		90 ± 20[δ]	< 0.2
TLT	65.7 ± 1.5	4M ⇌ 2D ⇌ T		70 ± 7[δ]	0.254 ± 0.005[δ]
CMI	97 ± 3	4M ⇌ T	49 ± 19	< 0.19[γ]	> 14[δ]
LMM	100 ± 4	4M ⇌ T	15 ± 12	< 0.14[γ]	> 7[δ]
IL	77 ± 2	2D ⇌ T		30 ± 6[δ]	< 0.2

[α] M = Monomer, D = Dimer, T = Tetramer.
[β] For comparison purposes; see Section 4.3.2.
[γ] Statistically significant greater tendency to form oligomer than wild-type.
[δ] Statistically significant smaller tendency to form oligomer than wild-type.

CD spectroscopy. Resulting data are shown in Figure 5.6. As discussed in Section 4.3.4, the data cannot be reliably fit to produce unique quantitative parameters. Qualitatively, the data support the basic analysis put forward in Section 4.4.2, and are consistent with the AUC results for each variant. AMAT, which was predominantly a mixture of monomer and dimer, also had the least intense baseline in low GdmCl. VVLM, which was essentially pure tetramer, had the most intense baseline. However, a direct correlation between the the baseline intensity and the species distribution was not possible for all variants. Although the first GdmCl-induced transitions for each variant varied considerably, the variants apparently lacking a dimeric state (ASAF, CMI, and LMM) had apparent transitions at the lowest concentrations of GdmCl. No variant was apparently more or less thermodynamically stable overall than is the wild-type receptor, as all

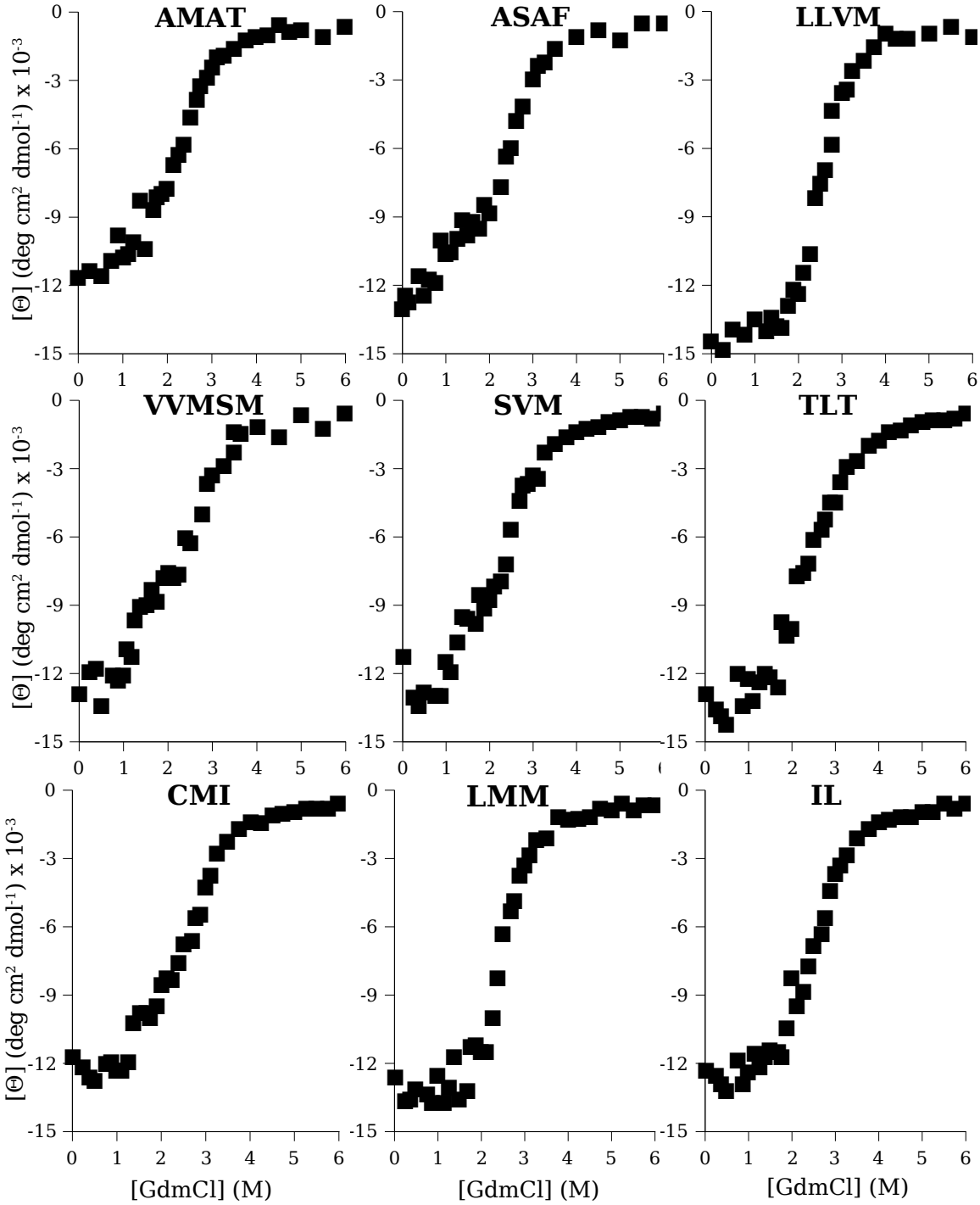

Figure 5.6. Guanidinium chloride induced unfolding of 10 µM hRXRα($D_{\beta}E$) variants. Monitored using CD spectroscopy at 222 nM.

variants were fully unfolded between 3 and 3.5 M GdmCl.

Table 5.9. Activity data for extra hRXRα(DβE) variants in yeast.

	9cRA[α]		LG335[α]	
Variant	**EC_{50} (nM)**	**Eff.[β] (%)**	**EC_{50} (nM)**	**Eff.[β] (%)**
I268M; I310V; F313L; L436A	> 10,000	0	60	70
I268A; A271S; A272S; I310M; F313L	> 10,000	0	100	100
I268V; A271S; I310V; F313V; L436V	> 10,000	0	200	80
A271S; A272T; I310S; F313L; L436A	> 10,000	0	200	40
I268A; I310A; F313A; L436F	> 10,000	0	220	70
I268A; A271S; I310V; F313V; L436V	> 10,000	0	300	140
I268V; I310V; F313S	> 10,000	0	440	10
I268A; I310S; F313V; L436F	> 10,000	10	470	60
I268L; A271V; I310L; F313L	> 10,000	0	530	20
I268A; A272V; I310A; F313A; L436F	> 10,000	0	530	30
I268L; I310M; F313V	> 10,000	20	610	20
I268V; A271V; I310L; F313V	> 10,000	0	650	10
I268L; I310V; F313I	> 10,000	0	2000	10

[α] All data from (221).
[β] Eff. = Efficacy, relative to wild-type hRXRα and 9cRA.

5.3.5 Prediction and Testing of Key Residues

A great aid to engineering functional variants would be the ability to identify individual mutations that are beneficial for the desired effect. Fox and coworkers recently reported on a straightforward approach based on linear combinations and partial least squares analysis (339-341). We applied this approach to a selection of variants to determine if the approach was useful for our system, with two important caveats: (i) the approach assumes the effects of mutations are essentially additive (i.e., independent and cumulative), which is reasonable for most proteins at most residues (342, 343) but has not been verified for nuclear receptors; and (ii) reliable analysis requires the ratio of data points (sequenced variants) to observed individual mutations should be at least 3. The ten variants in Table 5.2 (including wild-

Table 5.10. Predictions and actual effects of individual mutations.

		9cRA		LG335	
Variant	**Value Type**	**EC_{50} (nM)**	**Eff.[α] (%)**	**EC_{50} (nM)**	**Eff.[α] (%)**
Wild-type	Experimental	100	100	300	10
I268A	Predicted	1000[β]	30[β]	300	10
I268A	Experimental	1000	80[γ]	> 10,000[γ]	0[γ]
I310L	Predicted	100	20[β]	3[β]	90[β]
I310L	Experimental	> 10,000[γ]	0[γ]	> 10,000[γ]	0[γ]
I268A; I310L	Predicted	1000[β]	10[β]	3[β]	90[β]
I268A; I310L	Experimental	> 10,000[γ]	0[γ]	> 10,000[γ]	0[γ]

[α] Eff. = Efficacy.
[β] Predicted to be a change from wild-type.
[γ] Experimental result differs from prediction.

type) contain twenty mutations, for a ratio of 0.5. To improve the ratio, we discarded the CMI and IL variants, which contain several mutations not present in other variants, and added the additional variants shown in Table 5.9, for a total of 20 sequences and 26 mutations (0.77 ratio). See Section 5.2.7 for additional details of analysis.

Table 5.10 shows the tested predictions: the I268A and I310L mutations, and the variant containing both. The actual outcomes are also shown in Table 5.10, and deviate significantly from the predictions. The I268A variant had reduced 9cRA activity, but less than predicted, while the effect on LG335 activity was much stronger than predicted. The predictions for the I310L variant underestimated the actual effect on the 9cRA activity, and predicted precisely the opposite effect of the actual observations for LG335. The predictions for the double mutant were also quite different from the experimental results.

5.4 Discussion

5.4.1 Comparison to Prior Work

Relatively few reports on the biophysical characterization of RXRα variants exist. Of the work that has been performed, relatively little overlaps with the mutations in our variants. One prior report examined a large number of mutations in the 413-443 region of hRXRα, but only in the context of DNA binding (246). Another report indicated that activation function domain 2 is not essential for ligand binding (263). Many residues on the protein surface involved in homo- and heterodimerization have been mutated to alanine, with some mutants significantly disrupting self-association of mRXRα (277). The IL variant has been reported to have a lower limit K_d for 9cRA binding at least three times greater than the wild-type K_d (216), which agrees well with our observations (70 ± 20 nM for wild-type, 360 ± 80 nM for IL).

The F313A mutation (which is naturally occurring at the corresponding residue 318 in mRXRα), has been reported to reduce 9cRA affinity approximately 3-fold, while the F313I mutation has a smaller effect (254). The variants containing any of these mutations show a wide range of K_d values for 9cRA. F313A is also known to impair formation of the tetramer (262, 265). Our variants containing this mutation, AMAT and ASAF, had different self-association tendencies: AMAT is impaired in dimer, but not tetramer, formation, while ASAF has a tendency to form higher-order oligomers. These results, and the other variation in oligomeric behavior in our variants, suggest that oligomerization of hRXRα($D_{\beta}E$) is influenced by several binding pocket residues working in a non-additive manner, such that the mutations have a combined effect that is not the sum of the individual effects. This non-additivity has been previously observed in the residues that are part of the dimer interface: combining multiple alanine mutations

that individually disrupted dimer formation resulted in a variant with dimerization behavior similar to wild-type (277). The complete failure of our attempt to identify especially critical residues regulating ligand-dependent activity also suggests a high degree of non-additivity in binding pocket residue effects. The apparent relative lack of cooperativity in the GdmCl-induced unfolding also suggests a flexible structure that can adopt a range of energetically similar conformations.

Long-range effects of ligand binding have been reported in other nuclear receptors, such as ligand-specific effects on DNA and coactivator interactions (13, 14, 16, 17, 338, 344-353), oligomerization changes in RXRα (21, 201, 258, 265, 266, 294), and an example of binding of a ligand to RXRα inducing an active conformation in an unliganded heterodimer partner (354). The flexibility of nuclear receptors is demonstrated by the large changes in the ligand binding pocket to accommodate ligands (355-359), and is apparent from the NMR structures of hRXRα(E) (201) and peroxisome proliferator-activated receptor γ (360). More than half of the RXRα, RXRβ, and USP crystal structures contain large regions of very poor electron density (189, 190, 192, 194, 199, 200, 202-208), which indicates significant disorder; these are in addition to the first and last few amino acids that are not visible in any structures. Our results highlight that the binding pockets of nuclear receptors are complex regulatory mechanisms that trigger multiple, long-range allosteric effects upon ligand binding or introduction of mutations. At least two previous reports have attempted to identify the specific allosteric sites and "network" of residues allowing for the long-range effects (361, 362).

5.4.2 Correlations

As observed for wild-type hRXRα($D_{\beta}E$), the ΔT_m of the ligand-receptor pairs correlates quite well with the corresponding K_d values. Figure 5.7

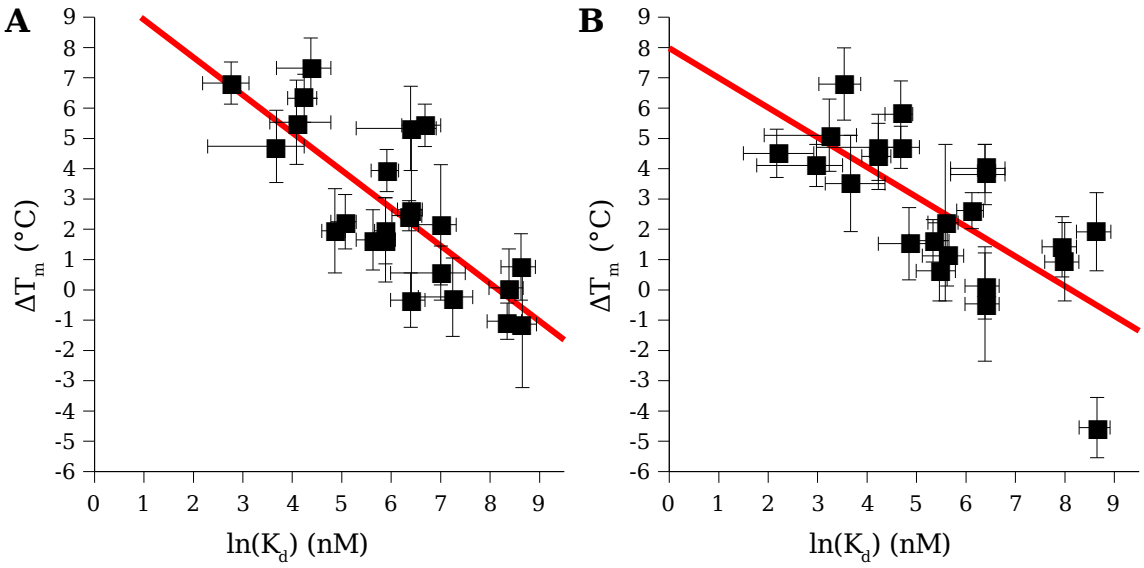

Figure 5.7. Linear correlations of thermal stability and ligand affinity for hRXRα(DβE) variants. Data includes the wild-type receptor. (A) Measurements in the absence of LxxLL peptide. (B) Measurements in the presence of LxxLL peptide. (Red lines, ▬) Linear fits.

shows the correlations, for values measured both in the absence and presence of the LxxLL peptide. The trends are significant: without the LxxLL peptide, $R^2 = 0.600$ and $p < 0.001$; with the peptide, $R^2 = 0.492$ and $p < 0.001$. These correlations suggest that the ligands serve as important structural elements in the receptors. Some of the scatter in the data is due to the comparison of kinetic (thermal denaturation) and thermodynamic (ligand binding)data, but the trend is still clear.

A significant correlation between activity of a protein and the binding affinity of a ligand has previously been reported (363), although the proteins involved were not nuclear receptors and the correlation only held over a narrow range of K_d. Another report demonstrated a strong correlation between ligand binding and heterodimerization (as a proxy for function) with the ecdysone receptor (364). A problem when attempting to correlate EC_{50} and K_d is that several receptors do not have a clear EC_{50} with some ligands (e.g., all variants except LMM with 9cRA). Although we did not

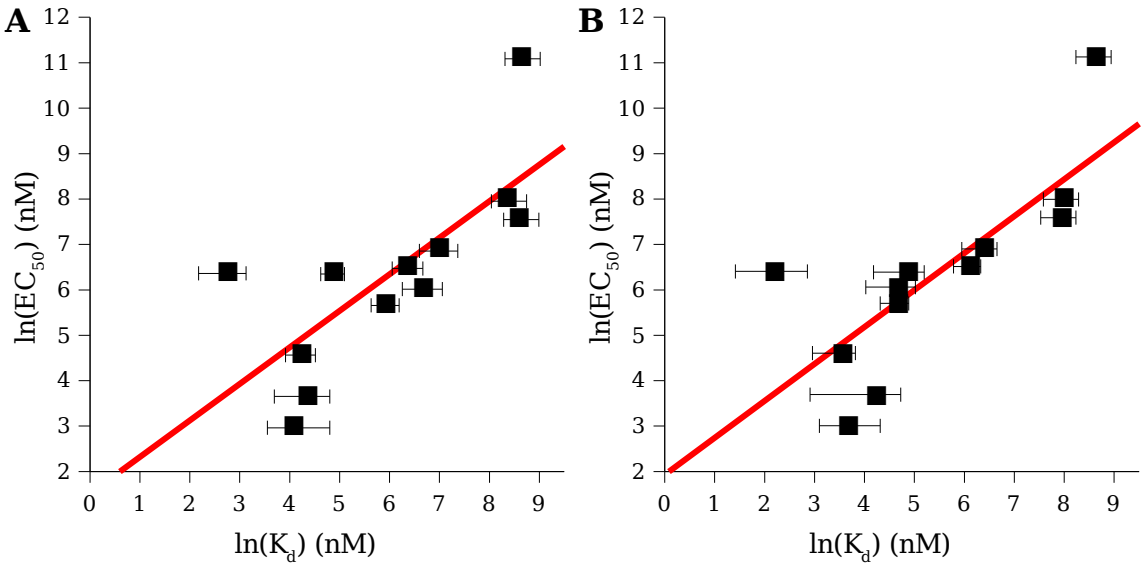

Figure 5.8. Linear correlations of EC_{50} and ligand affinity for hRXRα($D_{\beta}E$) variants. Data includes the wild-type receptor. Only ligand-receptor pairs where the ligand has clear agonist activity are included. (A) Ligand affinity in the absence of LxxLL peptide. (B) Ligand affinity in the presence of LxxLL peptide. (Red lines, ▬) Linear fits.

explicitly test for it, antagonism of any sort (i.e., a ligand that inhibits the receptor) would also be an exception to an EC_{50} vs. K_d correlation. An additional problem is that although preliminary results suggest that the concentration of LG335 in the medium (used to calculate the EC_{50}) is similar to that inside the cell (221), OPBA is significantly more soluble in the presence of cells than in buffer alone, suggesting absorption into the membrane and possibly an increase in intracellular concentration. Although the EC_{50} does have a statistically significant correlation with K_d when all data is included ($R^2 = 0.26$, $p = 0.02$), the trend is not strong.

An alternative approach is to exclude those ligands which do not have a clear EC_{50} value, which is equivalent to only using those ligands-receptor pairs in which the ligand is clearly an agonist. The correlation of K_d with EC_{50} for this subset is shown in Figure 5.8A ($R^2 = 0.550$, $p = 0.006$), which required discarding nearly half the data. Binding data obtained in the

presence of the LxxLL peptide, which might be expected to better model *in vivo* conditions, is not significantly better than the data in its absence (see Figure 5.8B; $R^2 = 0.594$, $p = 0.003$). These incomplete correlations suggest that additional processes are important in determining activity.

Efficacy does not correlate with K_d values, with or without the LxxLL peptide. The oligomeric distribution of the variants does not correlate well with any other property (K_d, T_m, EC_{50}, or efficacy).

Only K_d of only three variants changed in to the presence of the LxxLL peptide: CMI, IL, and ASAF. The lack of effect in most mutations may be related to a recent suggestion that ligands for RXRα may shift AF-2 into an optimal conformation (202). The specific mutations in the CMI and IL variants (which correspond to the residues present in human retinoic acid receptor γ) may be critical for LxxLL interactions. The mutations in CMI and IL are not present in the other variants except for I310M, despite the possibility for the F313I mutation in the library-selected variants. An additional indication of a crucial LxxLL interaction is that the IL variant had significant background (i.e., activity in the absence of ligand) activity in chemical complementation assays, but only when a human coactivator is used as part of the genetic selection system (218, 220). ASAF contains the naturally occurring F313A mutation, which has also been shown to trigger constitutive activity (254). Taken together, these results suggest that F313 may be crucial in ligand-coactivator allostery, and that the behavior is also modulated by other binding pocket residues.

The poor correlations between the ligand binding affinities and the activity (EC_{50} or efficacy) of the variants suggests that, in general, ligand binding is necessary but not sufficient for hRXRα activation. A corollary to this result is that selections or screens should operate directly on the property of interest (generally activity) rather than a proxy (such as ligand

affinity), because several properties interact to give rise to activity.

5.4.3 Implications

hRXRα has a highly flexible, dynamic structure. The inability to predict the effect of individual mutations, and the clearly non-additive behavior of multiple mutations, suggests that studies identifying specific mutations as having a particular effect are somewhat limited: a mutation may have a particular effect when it is the only mutation, but introducing a second mutation may cause radically different behavior from either mutation alone. As this non-additivity occurs through the LBD of hRXRα, all assignments of function to particular residues or regions may only be appropriate in a particular context (i.e., in the presence of other specific residues).

For future design or engineering of hRXRα, and likely nuclear receptors in general, the results we have presented are both a reason to be optimistic and pessimistic. On the pessimistic side, it appears that hRXRα is nearly the polar opposite of antibodies, which are essentially rigid bodies that recognize a ligand based on a relatively fixed shape. Simply "reshaping" the binding pocket of hRXRα to prepare it for a new ligand is unlikely to be successful, as the long-range effects of the binding pocket mutations will be significant. Our results also suggest that while the impact of individual mutations in a purely wild-type background can be characterized, the effects of the mutations will not be additive and the outcome of multiple mutations is not easy to predict

On the other hand, our results indicate that hRXRα is relatively tolerant of mutations and quite adaptable. These traits suggest that hRXRα can likely to be mutated to bind a wide variety of ligands while retaining function. The challenge will lie in creating an appropriate library to select the desired variant from, and in selecting directly for the desired property.

We suggest that libraries containing many sites (at least three) simultaneously mutated, rather than an incremental approach, is more likely to be successful, as mutations that are beneficial early in an iterative approach may prove to be detrimental later. In addition, primarily to maintain a reasonable library size, design of libraries should be targeted explicitly to the small molecule of interest, and only appropriate mutations allowed at each residue, such that the mutations complement the polar and hydrophobic regions of the small molecule.

CHAPTER 6
NONLINEAR DATA ANALYSIS

6.1 Introduction

Analysis of biochemical data by least-squares analysis is now quite common, and manuscripts have been published that detail the procedure and when it should be applied (312, 313). Least-squares provides a statistically rigorous method of obtaining a "best" fit to data, provided that certain criteria are met. The general formula for minimization using least-squares is given by Equation 6.1, which is the weighted sum of residuals squared:

$$\chi^2 = W \sum_{i=1}^{N} \left(\frac{Y(x_i) - f(x_i, a)}{\sigma_i} \right)^2 \qquad (6.1)$$

where W is a uniform weighting factor; N is the number of data points; $Y(x_i)$ is the Y value corresponding to the independent value x_i; $f(x_i,a)$ is the function being evaluated at point x_i using parameter set a; and σ_i is the standard deviation of the measured value $Y(x_i)$. χ^2 is evaluated in an iterative process, with the use of an appropriate algorithm to drive the fit toward a small value in each iteration. Alternative methods, such as robust parameter estimation (which uses a power other than two, or the absolute value, for the residuals), avoid some of the limitations of least-squares analysis, but are not as well characterized statistically (327, 365-367).

Perhaps the most commonly ignored criteria for using least-squares methods are: (i) the noise in the system must be Gaussian; and (ii) that all the uncertainty in measurements must be in the dependent (y) variables. Probably the most common technique of data fitting where least-squares is implicitly used incorrectly is when nonlinear data are linearized (312, 313, 324, 327), such as in Scatchard (325) or Cogan (326) plots. These types of

plots violate the principle of Gaussian noise by fitting the inverse of the data, which is a problem because a Gaussian distribution does not remain Gaussian when the inverse is taken. In addition, some of these linearizations divide the uncertainty in the measurements across both the dependent and independent variables. The net result is that nonlinear data fit by a linear method is statistically invalid, and in many cases is both quantitatively and qualitatively incorrect.

Numerous algorithms exist for generating a best fit. The most common when applying least-squares is the Gauss-Newton, often with variations introduced by Levenberg and Marquardt (313). This algorithm has several advantages: it is generally fast, converges quickly, is relatively easy to implement, and gives immediate error estimates using the final solution matrix. However, it has limitations: (i) the algorithm requires that the derivatives of the function being fit be calculated during the iterations, and (ii) the error estimation matrix generally produces errors that are too small (313). An alternative method, the Nelder-Mead simplex algorithm (306), is more general in that it does not require derivatives, but it is also somewhat slower. The Nelder-Mead simplex algorithm does not inherently provide any form of error estimation, but can easily be coupled to a rigorous method.

Numerous programs now exist for performing least-squares and other minimization routines. Here we present an outline of our approach, which makes use of a computer program, still under development, that is built to meet specific requirements: (i) perform reliable minimizations (fits); (ii) provide automated, rigorous error estimates; (iii) work regardless of the complexity of the function being fit (i.e., does not rely on calculation of a derivative); (iv) the ability to iteratively apply many models to many data sets automatically; (v) fit arbitrary combinations of data to an arbitrary

combination of functions in a global fit (e.g., denaturation data from fluorescence and CD spectroscopy simultaneously, which share a similar function but with some parameters specific to each data set); and (vi) modularity, such that the function being fit, the minimization algorithm (e.g., simplex), and the evaluation of the minimization (e.g., least-squares) could all be combined in any desired combination.

6.2 Theory and Program Description

Our customized program is coded entirely in Python. The use of this scripting language means that any part of the code may be quickly modified, and that the algorithms can be independently verified. In addition, the use of modules means that is simple to replace one algorithm or function for another, without rewriting large blocks of code.

As currently written, the program implements a variation of the Nelder-Mead simplex algorithm (306). The basic concept of a simplex is that for n parameters, there are $n+1$ vertices, where the coordinates for each vertex are guesses for the best-fit values of each parameter. In each iteration, the vertex containing the worst values (as evaluated by χ^2 or equivalent) is reflected through the center of all the other simplex points to generate a new value. This new value may be moved further in the same direction if it proves to be better than any existing vertex, moved partway back toward the center if it is worse than the vertex it originated from, or adjusted in several ways to ensure that no parameter is outside specified boundaries (e.g., to avoid providing the function with values that generate invalid results). The simplex size (distance between vertices) is dynamic during the fit, becoming smaller as the fit converges. In our case, we evaluate a fit as "converged" when the difference (in χ^2) between the best and worst vertex is below an arbitrary (very low) threshold. Regardless of the outcome of the placement of the new vertex (i.e., it is now the best,

worst, or an intermediate guess), the vertex that was second-worst prior to moving the worst vertex is now designated the "worst" (to avoid an endless loop of reflecting the same vertex).

Our modifications to the basic algorithm are slight. First, the rule of always moving the second-worst vertex following movement of the worst, even when the newly placed vertex is still worse than any others, means that the algorithm does not converge when there are only two vertices (one parameter). Therefore, we discard this particular rule for the special case of a two-vertex simplex. The second modification only comes into use when fitting multiple data sets simultaneously. Following a "global" simplex using all parameters, if groups of parameters exist that are specific to a subset of the total data, a "secondary" simplex using only these parameters and data sets will be performed while keeping all other parameters fixed. For example, if two data sets are being fit by six parameters, but parameters *A* and *B* are only relevant for fitting data set *1*, then a secondary simplex will be used to optimize parameters *A* and *B* while fixing all others, and ignoring data set *2* (which is irrelevant to the final values of *A* and *B*). This technique greatly increases the rate of convergence for large data sets and many parameters.

Our default evaluation scheme for minimization is least-squares, but that module can be replaced by an alternative method such as robust parameter estimation. In conjunction with an appropriately written function, additional algorithms such as numerical integration (368) could also be used.

In order to evaluate the quality of any fit, two elements are needed: (i) one or more measures of the "goodness" of the fit; and (ii) error estimates for the resulting parameters. Although the use of χ^2 in least-squares provides one measure of goodness and is useful to guide the fit toward

convergence, it does not ensure that the fit is appropriate. To provide an automated measure of appropriateness of the fit, we implemented residual analysis (311). The idea behind residual analysis is to ensure that the residuals of the fit are randomly distributed, as systematic deviations (such as most of the residuals being positive) can indicate non-random noise in the data or the use of an inappropriate model (function) being fit. In addition, statistical measures used to generate error estimates assume that the residuals are approximately Gaussian distributed (random). We have implemented the runs test, which is evaluated according to Equations 6.2 and 6.3:

$$E = \frac{2NP}{N+P} \tag{6.2}$$

$$Z = \left| (E-R) \frac{N+P}{(R-1)(R-2)} \right| \tag{6.3}$$

where E is the expected number of runs; N is the number of negative residuals; P is the number of positive residuals; and Z is the test statistic. A "run" is one or more continuous residuals with the same sign. Absolute Z scores of less than approximately 2 indicate a random distribution, while absolute scores of greater than approximately 4 indicate a major deviation. The runs test is more general than most alternative methods, but like all residual analysis methods it is of limited use when the number of data points is less than about 10.

Without a measure of the confidence of the final parameter values (error estimates), the fit is worthless. As with residual analysis, numerous methods exist for estimating errors, with a trade-off between reliability and calculation time (312). The fastest, and least reliable, is the use of the inverse of the final parameter matrix in Gauss-Newton methods. Similar matrix-based methods have been devised for the simplex algorithm,

resulting in essentially identical error estimates (369, 370). Matrix-based methods also assume symmetry in the errors (i.e., the errors for each parameter in the positive and negative directions are identical). Other methods rely on shifting parameter values, one at a time, to a fixed value away from the best fit, then allowing the new values to converge to a new "best fit." When the deviation in the parameter results in a statistically different fit, as evaluated by Fisher (F) and Student (t) statistics (371), the difference between the adjusted value and the original best fit value is the error estimate.

The most reliable method, and the most time-consuming, is a simple grid search, where each parameter is adjusted a small step at a time until a significant difference is found. Intermediate methods include the bootstrap (327, 365-367) and Monte Carlo (327, 372), each of which requires refitting the data hundreds or thousands of times. Non-matrix methods allow for dependency in the error estimates. Dependency is common in nonlinear fits, and arises when two or more parameters are correlated such that changes in one parameter can be somewhat compensated for by changing a second parameter (367, 373). Note that Fisher-based error estimation is only appropriate for least-squares based minimization; the bootstrap or Monte Carlo methods are more appropriate if using robust parameter estimation.

We have implemented a variation of the the grid scheme which provides errors of similar quality to the bootstrap and Monte Carlo methods, but generally does so at a smaller computational cost. In short, initial error estimates are found by moving the values for individual parameters in small, variable-sized steps until a significant difference is found in the fit, as evaluated using Fisher statistics:

$$F = \frac{\frac{\chi_2^2}{DF_2}}{\frac{\chi_1^2}{DF_1}} \tag{6.4}$$

where DF is the degrees of freedom (number of data points - number of parameters - 1); and $_1$ and $_2$ refer to the best fit and adjusted parameter values, respectively. The rapid initial estimate is done with all other parameters fixed, so provides an error estimate similar to that generated by the matrix-based methods. In a second round of estimation, each parameter is held at the value found in the first pass while the other parameters are refit to find the new minimum. If the new minimum is not statistically different from the the best minimum, the error estimate is increased by an amount proportional to the difference between the calculated F-value for the new minimum and the desired F-value for significance. This process is iterated until the new minimum is statistically different from the best fit, usually at a level of two standard deviations (p=0.05). The entire process, from initial estimate to final, is performed independently for both the negative and positive direction. Generally, the larger of the two resulting values is presented as the final error estimate (reduced to one standard deviation), although when this implied physically unrealistic results the asymmetric error is kept.

If parameters are not highly dependent, the error estimation process is rapid and generally only requires a few rounds; in the case of parameters with very high dependencies, our approach can take as long as a Monte Carlo or bootstrap method, and frequently results in asymmetrical errors. The difference between the initial and final error estimates gives the dependency:

$$D = 1 - \frac{E_1}{E_2} \tag{6.5}$$

where D is the dependency; E_1 is the initial error estimate; and E_2 is the final error estimate. A dependency of less than 0.5 (i.e., less than two-fold difference between the two error estimates) is excellent, while dependencies of greater than approximately 0.9 (i.e., greater than ten-fold difference between the two error estimates) suggest that the number of parameters should be reduced or that additional data may be needed to generate a reliable fit (367, 373).

A critical issue in all nonlinear fits is whether the true minimum has been found. One advantage to non-matrix based error estimates is that the local space around the best fit value is thoroughly investigated; if during the course of the error estimation a fit is found that is superior to the previously calculated best fit, the new fit is taken as the starting point and the entire process is restarted. For complex functions, this reoptimization is useful to ensure the global minimum is found. In addition, multiple fits should be performed using different starting guesses; different initial guesses leading to the same final fit is suggestive of a global minimum. Our code automates the process of starting from a variety of random initial guesses, and stores the desired number of resulting fits to perform thorough error estimates on all of them. For simple functions, the entire procedure is not necessary, but is very rapid if performed anyway.

As shown in Equation 6.1, weighting can be used in two ways during a least-squares fit; its use in other methods may be similar but is not as statistically sound (327). The first method is to weight each residual by the uncertainty in the measurement (σ_i), which propagates the uncertainty of measurements into the fit and gives more reliable parameter estimates (374). The second method is to weight all data points by a fixed value (W), which is equivalent to multiplying the final χ^2 by a fixed value. Fixed value weighting is useful in two ways when performing global fits: (i) a scalar can

be used to correct for large differences in the magnitudes of the dependent variables to avoid domination of χ^2 by a single data set; and (ii) to correct for the number of data points, as larger data sets tend to dominate χ^2. The dependence of χ^2 on the number of data points is usually desired, but in cases where equal weighting of data sets is desired, Equation 6.6 can be used:

$$W = \frac{1}{N^2} \tag{6.6}$$

where N is the number of data points. All weighting methods are easily automated.

The program allows for complete control over the fitting behavior, including details such as the convergence threshold, maximum number of iterations, and weighting. In addition, the algorithm allows for two types of constraints: (i) absolute constraints, which the parameter value may never exceed (useful to avoid providing an unacceptable value to a function); and (ii) partial constraints, which are used to constrain the initial random guesses and guide the fit, but which may be exceeded to estimate errors and at other specific times.

6.3 Applications

All applications are taken from actual analysis in Chapters 4 and 5. Therefore, only the theoretical justification and relevant program details are provided here.

6.3.1 Ligand Binding Data

The ligand binding data presented in Sections 4.3.1 and 5.3.1 do not meet the least-squares criteria of having Gaussian noise, because the division of two Gaussian distributions does not produce a Gaussian distribution. Therefore, the data were fit using both least-squares and

robust parameter estimation (using the absolute value of the residual instead of its square). However, no significance difference was found in the outcome of the fits, so we ultimately used exclusively the least-squares algorithm. The lack of a significant difference probably occurs because the noise in the binding model is approximately Gaussian.

The use of the simplex algorithm produced identical best-fit values as the use of the Levenberg-Marquardt algorithm, but not identical error estimates and dependencies. The magnitude of the differences vary somewhat with individual data sets, but matrix methods produced errors similar to or somewhat smaller than the simplex initial error estimates. Matrix methods also overestimated the dependencies, with values of 0.6-0.8, while the Fisher-statistics method resulted in dependencies of 0.3-0.5. The K_d errors are asymmetric, with the error in the positive direction being approximately 2.5-fold larger than the error in the negative direction.

6.3.2 Thermal Denaturation Data

The thermal denaturation data presented in Sections 4.3.3 and 5.3.2 was globally fit to a single function and set of parameters. The function did not always converge directly to the global minimum, so many initially guesses were used. Approximately half the initial guesses led to a best fit, so the best fit from ten initial guesses gave a reproducible set of final parameter values. The parameters have dependencies of up to 0.9, so error estimates are a much better measure of the true uncertainty in the parameter values than from matrix methods.

6.3.3 Analytical Ultracentrifugation Data

The analytical ultracentrifugation data presented in Sections 4.3.2 and 5.3.3 were fit using a single function and several data sets. One or more parameters was only relevant to a single data set, and so secondary simplex

optimization helped the fit converge to a global minimum. Several initial guesses and full error fitting of two or more "best fits" was necessary to verify the global minimum. The parameters are highly correlated, usually with dependencies greater than 0.9.

Prior work has demonstrated that the noise in AUC is not Gaussian (365-367), and therefore least-squares analysis is not appropriate. However, for our data, alternative analyses based on robust parameter estimation did not produce quantitatively different results.

Several programs exist that are dedicated to the fitting of analytical ultracentrifugation data, including Sedfit/Sedphat (317), UltraScan II (375), and SedAnal (http://sedanal.bbri.org/). However, the programs are limited by the inability to automatically generate reliable errors, and/or the inability to automate application of multiple models to many different data sets. As some models took hours to fit (in our program and others) automatically fitting the data from multiple experiments to multiple models overnight was extremely useful. On the other hand, our program currently lacks a nifty graphical interface.

6.3.4 Chemical Denaturation Data

The denaturation data described in Sections 4.3.4 and 5.3.4, makes use of functions that cannot be written without the use of a programming language, and cannot be fit using algorithms requiring a derivative (e.g., Levenberg-Marquardt). A derivative for quartic Equation 4.23 cannot be computed, as the dependent variable cannot be isolated. "If" statements and other programming controls are necessary to avoid errors arising from rounding in floating point values. Equations 4.22 and 4.23 assume that all four possible species contribute significantly to the signal; when the fit parameters were evaluated for a particular data point and a species was found to represent less than 0.001% of the total concentration, the

appropriate term was dropped and the data point re-evaluated repeated using the resulting simpler model.

The use of rigorous statistical analysis led us to conclude that the range of statistically equivalent fits for this data was too large to present a reliable set of parameters. Matrix-based approaches error estimates did not indicate the great uncertainty. Despite numerous attempts to eliminate (keep fixed) parameters during the fits, especially slopes and intercepts for the individual species baselines, nothing was sufficient to yield statistically valid, unique parameter sets.

6.4 Discussion

In summary, our approach to, and program for, nonlinear analysis provides reliable error estimates, the ability to fit any function using a variety of minimization functions and algorithms, automation, and complete control over the fitting process. At the time of this writing, the program is not yet in a state ready for distribution to and use by the general scientific community, but a general cleanup and effort to make it user-friendly is in progress; when this process is complete, it will be made freely available.

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this work, we have addressed several aspects of engineering nuclear receptors, from how to identify a unique genomic DNA target sequence and construct the appropriate DNA binding domain, to rigorous analysis of variants and possible mechanisms of nuclear receptors function. hRXRα is a slightly flexible cavity that lock onto ligands, but rather a truly dynamic protein with numerous long-range, allosteric sites, capable of significant structural change in response to slight stimuli and of exquisite ligand discrimination. The functional diversity of hRXRα (and nuclear receptors in general) can't easily be explained by a relatively static structure.

Our sample size of variants is far too small to provide true guidelines as to how to select residues for mutation in hRXRα. However, our results do illustrate that explicit protein design of hRXRα will most likely fail, given our current level of ignorance: the high degree of interaction between residues and the diverse long-range effects defy simple predictions. Studying single amino acid mutations one at a time may be misleading when trying to identify the function of specific residues, as a second mutation may well offset the effect of the first. On the other hand, the hRXRα is clearly quite adaptable, and apparently tolerates diverse multiple mutations. This behavior suggests that, given an appropriate library, an hRXRα variant can be identified to bind and be activated by an arbitrary small molecule. However, our data also suggest that selection should be performed directly on the property of interest, as hRXRα may be able to adapt to a proxy property without having any affect on the property of interest.

7.2 Future Work

Certainly the most requested experiments, and those with the greatest unknown as a time investment, are those involving more structural characterization. The most ambitious goal would be crystallization of the variants described above, with and without ligands and the LxxLL peptide, to try to correlate changes in the variants' properties with changes in their structure. However, crystallization of nuclear receptors is not trivial; despite thousands of trials, we were unable to replicate existing reports by crystallizing wild-type hRXRα($D_{\beta}E$). Moreover, our data suggests that crystal structures may not be as informative as generally assumed due to their inherently static nature. NMR studies would potentially be more useful, as the measurements include solution dynamics, but the fact that very few nuclear receptors have been studied by this technique suggests the difficulty. We were unable to prepare samples containing sufficiently concentrated hRXRα($D_{\beta}E$) for NMR experiments without using very high salt concentrations or other interfering additives. Computational analysis of the variants using known structures as models may also provide additional insights.

Other structural techniques that lack atomic-level detail might also be useful. Although we are unable to probe the effects of ligand binding on oligomerization via analytical ultracentrifugation, and techniques such as dynamic light scattering and gel filtration chromatography require assumptions of size and shape, other techniques can avoid these problems and assumptions. Atomic force microscopy, small angle x-ray scattering, and neutron scattering can provide direct measurements of oligomer dimensions, which would be useful for eliminating assumptions of size and shape. These techniques might also be applicable to full-length receptors, which are apparently not amenable to crystallization.

Experiments that could provide comparable information to the structural studies are those probing the dynamics and kinetics of hRXRα. We have not examined the kinetics of ligand binding to variants or wild-type hRXRα($D_{\beta}E$) at all, although such studies might provide insight into the weak correlation between binding and activation. Probing the dynamics of the variants, via techniques such as hydrogen-deuterium exchange or ICAT labeling of MPAX-prepared (323) variants, might also provide key insights into both structural and dynamic information related to ligand binding. We have generated hRXRα variants with three of the four cysteine residues removed to allow MPAX-based labeling, and verified that they express well, but were unable to successfully label and analyze them using the MPAX-ICAT system. Two residues were mutated to consensus amino acids (C369L and C404A), and a third with a mutation that has previously been shown not to affect function (C432G). The final cysteine, C269, was never visible by mass spectrometry analysis, and so we did not replace it, although in retrospect its presence may still be causing experimental difficulties (by consuming labeling reagent and occupying most of the affinity sites during ICAT post-labeling cleanup).

To possibly resolve the difficulties with analyzing the GdmCl-induced unfolding data collected by CD spectroscopy, AUC experiments using buffers containing increasing amounts of guanidinium chloride could be performed and the data globally analyzed with existing CD data. Although we performed preliminary experiments on the effects of salt and pH on the apparent T_m, we have by no means exhausted such studies. The use of more realistic salt and buffer conditions (in intracellular terms), in combination with conditions mimicking cellular crowding such as high concentrations of glycerol or bovine serum albumin, may provide insight to the actual oligomerization and aggregation behavior of hRXRα within cells. Similar

experiments could be performed to examine the effects of such conditions on the ligand binding. The effect of temperature on the ligand binding and oligomerization might provide further insight into the thermodynamics of both processes.

A more ambitious project would be examine the effects of the known, and possible, post-translation modifications of RXRα on the protein's behavior. A different (possibly larger) coactivator fragment could be used to see if the effect on ligand affinity is common to all motifs, and/or to include a fluorescence amino acid or label to enable direct measures on coactivator affinity; similar experiments could be done with corepressor motifs.

The obvious future work is to simply repeat all the described experiments on more variants, of which many exist. However, one crucial open question is the cause of non-activation in non-functional variants. Quite possibly many simply do not express or fold well, but it would be interesting to characterize a handful of variants that appear to express and fold but remain inactive. While we hope that our results begin to provide an explanation for why variants behave as they do, it may be that the non-functional variants show a similar range of behavior, and the real cause of the changes in function have to do with properties that have not yet been measured.

The additional wild-type hRXRα constructs (described in Chapter 3) would be interesting to follow up on and data to compare to hRXRα($D_{\beta}E$). Nuclear receptors are treated as modular, but a systemic study of ligand binding, oligomerization, thermal stability, and thermodynamic stability for hRXRα($D_{\beta}E$), hRXRα(CDE), hRXRα(ABCDE), and GBDhRXRα(DE) would be an excellent test of the limits of modularity. The GBDhRXRα(DE) construct might be purified in *E. coli* with relatively little effort, by a creative use of affinity tags or other purification techniques to get around the protease

cleavage problem. hRXRα(CDE) may behave better with the C195A mutation introduced in the studies studying hRXRα(C) (212-214, 255); personal communications also suggest that the zinc concentration needs to be kept low and tightly controlled. It is likely that hRXRα(ABCDE), as well as hRXRα($ABCD_{\alpha}$), will need to be expressed using a somewhat different system: either a large tag such as maltose binding protein, an intein expression system, or in some eukaryotic organism. Any constructs containing the DBD also introduce the possibility of performing experiments in the presence and absence of DNA.

Probably the two most ambitious future projects also involve the greatest payoff if they are successful. A systematic generation of mutations in all binding pocket residues in RXRα, and characterization of those variants, would allow generation of a complete picture of the long-range interactions involving the binding pocket and the coupling between residues. Although such a complete replacement throughout the binding pocket of even the single amino acid mutations would require several thousand variants, such a study need not be quite so exhaustive to provide useful information. However, lacking high-throughput techniques for purification and characterization, such a project is likely too ambitious for the present. A more focused and manageable project might be to test the predictions of networks of interacting residues or other previously reported "key" positions, not only to determine if the ideas are valid but also to examine how well such a set of interacting positions holds up when multiple mutations are made.

A second high payoff project would be to replicate comparable studies using a different nuclear receptor. As different nuclear receptors will probably require re-optimization of every step of the work detailed above, such a project is not lightly undertaken. Moreover, RXRα is convenient

because it contains two tryptophan residues in the binding pocket, enabling the fluorescence assay for ligand binding; a tryptophan may not be so readily available in other nuclear receptors. Although challenging, expression of multiple nuclear receptors not only opens doors for studies on new receptors, but also on interactions between receptors. A somewhat easier, yet still valuable, project would be to examine hRXRβ and hRXRγ, which are orthologs of hRXRα and so might provide additional insights.

One of the primary goals of the work described here is to guide future protein engineering on RXRα, with the goal of enabling the applications discussed in Chapter 1. To that end, the long-range work that still needs to be accomplished is to design not only a functional ligand binding domain, but also to couple it to an experimentally validated arbitrary DNA binding domain, as described in Chapter 2. We have provided a guide to designing and analyzing rigorous characterization, designing the necessary DNA binding domain to target the variant as necessary, and suggestions for designing libraries that will yield variants with the desired function.

APPENDIX A
ESPSEARCH CODE

```
# ESPSearch.py, version 1.01
""" Searches DNA sequence for binding sites and analyzes for patterns.
    Copyright (C) 2004-2005 Terry J. Watt and Donald F. Doyle
    This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
    You may contact Dr. Donald Doyle at:
    donald.doyle@chemistry.gatech.edu
    School of Chemistry & Biochemistry
    Atlanta, GA 30332-0400
    See espsearch.html for instructions on using this script and for the GPL.
    If publishing any results obtained with this program, please cite:
    Watt, T. J. and Doyle, D. F.  ESPSearch: A Program for Finding Exact
        Sequences and Patterns in DNA, RNA, or Protein.  Biotechniques, 2005,
        38, 109-115.
    See http://web.chemistry.gatech.edu/~doyle/espsearch/ for more information.
"""
#--------------------------------------------------------------------------
def AbortProgram(ExitString):
    """Ends the program, printing any necessary messages.
    """
    import sys # to enable forced exit
    print ExitString # exit message
    ExitVar = raw_input('Press <ENTER> to exit.') # wait for input before exit
    sys.exit() # force program exit
#--------------------------------------------------------------------------
def DefinePattern(PatternToMatch, ShortestTarget, LongestTarget, Mode, DatabaseFrag):
    """Generates lists that represent the user-specified patterns.
    This routine cycles through the user-entered pattern(s) to create a
    list of rules and other useful information for each pattern.  Everything
    is ultimately stored in PatternRestrictions and returned.
    PatternRestriction is arranged as follows:
    [x][0] = Whether or not this pattern contains any unique positions
    [x][1] = Pattern as entered by user
    [x][2] = Output file name associated with the pattern
    [x][3] = Pattern in segments for generating header info in output file
    [x][4] = All Pattern information, by position (letter or gap)
    for non-gap positions:
    [x][4][y][0] = # of nearest previous position that is identical (-1 if none)
    [x][4][y][1] = Strand (1 if complement (<), -1 if primary (>), 0 if either (|));
```

```
always -1 for proteins
    [x][4][y][2] = Indicator of unique position if 1 (!)
    for gap positions:
    [x][4][y][0] = Low end of gap range
    [x][4][y][1] = High end of gap range
    """
    # first determine if any patterns even exist to parse
    if len(PatternToMatch) == 0: return [], LongestTarget # the minimum information for
the rest of the routines to function smoothly
    PatternChars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz' # allowed chars
for pattern
    Digits = '-0123456789' # allowed values for gaps
    PatternRestrictions = []
    MinPatternBuffer = LongestTarget # for determining how many positions need to be
remembered in buffer
    MultPatterns = PatternToMatch.split(',') # break into as many patterns as exist
    CurPatNumber = 0 # for generating file names
    for EachPattern in MultPatterns: # for each pattern
        CurPatNumber = CurPatNumber + 1
        NumLetters = 0 # number of non-gap positions in pattern
        AnyUnique = 0 # if any positions are unique, will get tagged to -1
        EachPattern=EachPattern.strip() # remove whitespace
        PatternFile = '%sp%s' % (DatabaseFrag, CurPatNumber) # for
PatternRestrictions[x][2]
        PatternList = [] # for PatternRestrictions[x][4]
        HeaderList = [] # for PatternRestrictions[x][3]
        MaxGapLength = 0 # for determining buffer length
        PatPos = 0 # position in pattern while parsing
        CurStrand = -1 # var to interpret >, <, and | in pattern, default is >
        Uniqueness = 0 # var to interpret unique position, ! in pattern
        if EachPattern[-1] == ']': # if pattern ends with gap, exit
            AbortProgram('Error: terminal gap found in pattern.')
        if EachPattern[0] == '[': # if pattern begins with gap, exit
            AbortProgram('Error: pattern begins with a gap.')
        while PatPos < len(EachPattern): # not for loop b/c "step" is variable
            if EachPattern[PatPos] == '[': # if position indicates the beginning of a gap
                PatternFile = PatternFile + '_' # indicate gap in file name
                # now search forward for end of gap (])
                for GapEnd in range(PatPos+1, len(EachPattern)):
                    if EachPattern[GapEnd] == ']':
                        break
                    elif EachPattern[GapEnd] in PatternChars: # if found letter before ],
exit
                        AbortProgram('Error: missing closing ] on gap in pattern.')
                CutStr = EachPattern[PatPos+1:GapEnd] # the isolated gap w/o brackets
                RangeVars = CutStr.split(':') # now split into #s
                if len(RangeVars) != 2: # if contains more or less than 1 ":"
                    AbortProgram('Error: pattern gaps must be in the format [X:Y], where
X & Y are integers.')
                else:
                    for RangeEntry in RangeVars: # determine if any value is empty or
invalid
                        if len(RangeEntry) == 0 or RangeEntry.strip(Digits) != '':
                            AbortProgram('Error: pattern gaps must be in the format
[X:Y], where X & Y are integers.')
```

```
                if min(int(RangeVars[0]), int(RangeVars[1])) < -ShortestTarget: #
determine if negative value is too negative
                    AbortProgram('Error: magnitude of negative number in gap is larger
than shortest possible target.')
                PatternAppend = [min(int(RangeVars[0]), int(RangeVars[1])),
max(int(RangeVars[0]), int(RangeVars[1]))] # put smaller value first, larger second
                MaxGapLength = MaxGapLength + PatternAppend[1] # increase cummulative
maximum gap length
                HeaderList.append(EachPattern[PatPos:GapEnd+1]) # store gap info for file
header
                PatPos = GapEnd # move position to end of gap
                PatternList.append(PatternAppend) # add new info
            elif EachPattern[PatPos] == '>': # following positions are on primary strand
                CurStrand = -1
            elif EachPattern[PatPos] == '<': # following positions are on complement
strand
                CurStrand = 1
            elif EachPattern[PatPos] == '|': # following positions could be on either
strand
                CurStrand = 0
            elif EachPattern[PatPos] == '!': # following position is unique
                Uniqueness = 1
                AnyUnique = -1
            elif EachPattern[PatPos] not in PatternChars: # invalid character, exit
                AbortProgram('Error: invalid character (' + str(EachPattern[PatPos]) + ')
in pattern.')
            else: # a letter
                NumLetters = NumLetters + 1
                # adjust header output appropriately for strand & store
                if Mode == 1:
                    PatternAppend = [-1, CurStrand, Uniqueness] # rules for this position
                    if CurStrand == 1:
                        HeaderList.append('<' + EachPattern[PatPos])
                    elif CurStrand == 0:
                        HeaderList.append('|' + EachPattern[PatPos])
                    else:
                        HeaderList.append('>' + EachPattern[PatPos])
                else:
                    PatternAppend = [-1, -1, Uniqueness] # rules for this position
                    HeaderList.append(EachPattern[PatPos])
                PatternFile = PatternFile + EachPattern[PatPos] # update file name
                if Uniqueness == 0: # if not a unique position, identify previous
position that is same (if any)
                    for ListPos in range(len(PatternList)-1, -1, -1):
                        if EachPattern[PatPos] == PatternFile[ListPos]: # found position
of same type, mark current position to point to it
                            PatternAppend[0] = ListPos
                            break # only want closest position, so stop now
                Uniqueness = 0 # now reset Uniqueness indicator to default
                PatternList.append(PatternAppend) # add new info
            PatPos = PatPos + 1 # move to next position
        # now append pattern information to master list & determine buffer needs
        PatternRestrictions.append([AnyUnique, EachPattern, PatternFile, HeaderList,
PatternList])
        LenPattern = NumLetters*LongestTarget + MaxGapLength
```

```
        MinPatternBuffer = max(MinPatternBuffer, LenPattern)
    # add .txt extension to all file names
    for EachPattern in range(len(PatternRestrictions)):
        PatternRestrictions[EachPattern][2] = PatternRestrictions[EachPattern][2] +
'.txt'
    return PatternRestrictions, MinPatternBuffer
#---------------------------------------------------------------------------
def DoOSOperations(InputFiles, OutputDirectory):
    """Creates directories & tests write ability.
    If successfully writes to assigned output directory, creates new
    directories to store output.  New directories are named after the input
    file, to allow multiple source files in a single run of this script.
    """
    import os # necessary to delete files and create directories
    # test to see that can write to output directory, if not then exit
    TestWriteFile = OutputDirectory + 'espstemp.txt'
    try:
        TestFile = open(TestWriteFile, 'w')
    except IOError:
        AbortProgram('Error: output directory does not exist or cannot be written to.')
    TestFile.close()
    os.remove(TestWriteFile) # delete temporary file
    # now know that output directory exists and is writeable, so create directories
    NewDirectories = []
    # name directories after input file, w/o extensions
    for FileNum in range(len(InputFiles)):
        InputFrag = InputFiles[FileNum][:]
        for SlashPos in range(len(InputFrag)-1, -1, -1):
            if InputFrag[SlashPos] == '/':
                InputFrag = InputFrag[SlashPos+1:] # remove / and everything before
                break # because only looking for last one
        for DotPos in range(len(InputFrag)-1, -1, -1):
            if InputFrag[DotPos] == '.':
                InputFrag = InputFrag[:DotPos] # remove . and everything after
                break # because only looking for last one
        NewDirectories.append(OutputDirectory + InputFrag + '/')
        # allow for possibility that the "new" directory already exists
        try:
            os.mkdir(NewDirectories[FileNum])
        except OSError:
            # hopefully just means already exists
            pass
    return NewDirectories
#---------------------------------------------------------------------------
def FindSequences(SeqMatches, InputFileName, OutputDirectory, TargetLength,
MaxMismatches, FileNameArray, PatternRestrictions, OutputType, PatternType, BufferSize,
MinTarget, MinBufferLen, PatternQueueLen, ComplementKeys, Mode):
    """Searches a source sequence for the desired sequences & outputs results.
    First prepares all the output files and initializes a bunch of variables.
    Does a bunch of processing to handle header info in files and loading of
    buffers.  Reads in a buffer of positions from source file to optimize
    performance elsewhere; not using a buffer of reasonable length (500+)
    means that the time spent on things besides searching becomes significant.
    To identify target sequences, looks at a "window" (determined by shortest
```

```
    end segment in database) in buffer; if that sequence exists in SeqMatches,
    then one or more targets match it.  For output type 2, this immediately
    leads to output, because only 1 segment in each target.  For others,
    need to determine 1) if any hits at this position are the first in a target,
    2) if any "extra" positions (if segment is longer than shortest) match
    the nex ones in buffer, 3) and find other segments for this hit by
    reading ahead in buffer the appropriate distances (defined by gaps) to see
    if the sequences match.  When output is generated, hits are stored in
    CompleteHitQueue for later processing by patterns.
    Patterns are identified in a similar fashion, but a longer buffer is needed,
    so patterns keep a second buffer current, updated by the main buffer.
    In principle they work the same way as a target with multiple segments,
    but it is a little more complicated because it can't be assumed that each
    segment of a pattern is separated by a gap, patterns can cross strands,
    and patterns can have unique positions.  Gaps are handled by treating them
    explicitly (not implicitly as in targets), and cross strand behavior by
    creative use of multiplication.  Unique ones are handled immediately before
    output.  Patterns do have it easier in that everything stored in CHQ is a
    hit, so processing can actually be faster for searches involving gaps than
    with target sequences.  Accessing information in CHQ is also much faster
    than SeqMatches, because it is typically a much smaller dictionary.
    Though certain aspects of this code may not look optimized, the majority of
    time is usually spent on dictionary lookups (mostly generating the hash
    values), which is a more or less fixed cost somewhat influenced by
    dictionary size.
    """
    # CHQ[x] = all hits for given position value (x based on output position value)
    # CHQ[x][y][0] = sequence (for +)
    #          [1] = strand for hit
    #          [2] = hit name
    #HitList[x] = data for part or all of successful hit
    #       [x][0] = Start of sequence (point to buffer position)
    #       [x][1] = minimum distance of current gap dealing with
    #       [x][2] = maximum distance
    #       [x][3] = Current mismatch count
    #       [x][4] = list of gap values used to create this hit
    #       [x][5] = length of next segment
    # Patternhits[x] = partial (until complete) pattern
    #            [x][0] = next position in CHQ to examine
    #            [x][1] = list of dictionaries/lists of hits/gaps (1 for each pattern
entry)
    #            [x][2] = list of sequences for pattern positions (for sequence-level
patterns)
    NumberPatterns = len(PatternRestrictions)
    InputFile = open(InputFileName, 'r') # open the input file
    # generate appropriate output files for given output type
    OutputFile = []
    for FileNum, FileName in enumerate(FileNameArray):
        OutFile = OutputDirectory + FileName
        try:
            OutputFile.append(open(OutFile, 'w'))
        except IOError:
            if FileNum == 0: # if failed on first file
```

```
                AbortProgram('Error: output directory does not exist or could not be
written to.')
            else: # failed because too many open files
                for EachFile in OutputFile:
                    EachFile.close()
                AbortProgram('Error: too many files generated for operating system.
Change output format or reduce number of target names.')
    PatternFile = {} # for quickly accessing which file to write to for patterns
    # open any files necessary to write out pattern results
    for EachPattern in PatternRestrictions:
        PatFile = OutputDirectory + EachPattern[2]
        try:
            PatternFile[EachPattern[1]] = open(PatFile, 'w')
        except IOError:
            # most likely b/c too many open files
            AbortProgram('Error: failed to create pattern file, possibly because too many
files are open.  Change output format or reduce number of target names.')
    if OutputType != 2:
        # output header information into all files
        if Mode == 1:
            OutputString = '%sPosition\tSequence\tStrand\tMismatches%s\n' %
('Target\t'*OutputType, '\tGaps')
            for PatNum in range(NumberPatterns):
                PatternFile[PatternRestrictions[PatNum][1]].write('Position\tSequence\tSt
rand%s\n' % (''.join([('\t' + PatHeader) for PatHeader in
PatternRestrictions[PatNum][3]])))
        else:
            OutputString = '%sPosition\tSequence\tMismatches%s\n' %
('Target\t'*OutputType, '\tGaps')
            for PatNum in range(NumberPatterns):
                PatternFile[PatternRestrictions[PatNum][1]].write('Position\tSequence%s\n
' % (''.join([('\t' + PatHeader) for PatHeader in PatternRestrictions[PatNum][3]])))
        for FileNum in range(len(FileNameArray)):
            OutputFile[FileNum].write(OutputString)
    else: # OutputType = 2
        # adjust headers for Target size and output
        if Mode == 1:
            OutputString = 'Position\tBase%s' % (''.join([('\tFrame' + str(FrameNum+1))
for FrameNum in range(MinBufferLen)]))
            OutputFile[0].write(OutputString)
            OutputFile[1].write(OutputString)
        else:
            OutputFile[0].write('Position\tAmino Acid')
        # output header information into pattern files
        for PatNum in range(NumberPatterns):
            PatternFile[PatternRestrictions[PatNum][1]].write('Position\tSequence\tStrand
%s\n' % (''.join([('\t' + PatHeader) for PatHeader in PatternRestrictions[PatNum][3]])))
    # ensure buffer is long enough to hold info for longest possible target & pattern
    if BufferSize < MinBufferLen:
        BufferSize = MinBufferLen + BufferSize
    if BufferSize < PatternQueueLen:
        BufferSize = PatternQueueLen + BufferSize
    # Initialize all the variables necessary to keep track of search
    SequenceInfo = '' # for separating sequences in output
    NewSequenceInfo = '' # for storing the next sequence header when reach end of a
```

```
sequence
    StripChars = '0123456789 \n\t' # for removing whitespace & line numbers
    MinListLen = 1-MinBufferLen # for shifting buffer
    MinQueueLen = 1-PatternQueueLen # for shifting pattern buffer
    Sequence = '' # will hold the current "window" being examined
    Buffer = '' # holds the current set of positions being scanned
    CurrentLine = '' # for processing each line as read in
    CurrentInLine = '' # for processing each line as read in
    BeginSeq = 1 # for triggering behavior needed at beginning of sequences
    OutputFirstBases = 1 # for triggering initial output needed for Output Type 2
    FirstSeq = 1 # for triggering special behavior for the very first sequence
    EOF = 0 # for knowing when file is done being read
    FirstHitRef = [1 for FileNum in range(len(FileNameArray))] # for regenerating list of
whether output sequence info into files
    FirstPatHitRef = [1 for PatNum in range(NumberPatterns)] # for regenerating list of
whether output sequence info into pattern files
    # find hits
    while EOF == 0: # until eof
        # do processing for beginning of sequence
        if BeginSeq == 1:
            FirstHit = FirstHitRef[:]
            FirstPatHit = FirstPatHitRef[:]
            if OutputType == 2: # output last few positions to sense strand/protein
sequence
                if Buffer != '': # max in next line used in case sequence is extremely
short
                    OutputFile[0].write(''.join(['\n%s\t%s' % ((LowCounter+BufferPos+1),
Buffer[BufferPos]) for BufferPos in range(max(len(Buffer)+MinListLen, 0), len(Buffer))]))
                OutputFirstBases = 1
            Adjustment = MinListLen # for knowing how much of buffer to save between
loads
            PatternAdjustment = MinQueueLen # for knowing how much of pattern buffer to
save
            BeginSeq = 0 # no longer the beginning of a sequence
            MaxCounter = 0 # reset all counters (specialized counters are based on this
one)
            CompleteHitQueue = {} # reset remembered hits for patterns
            CHQBuffer = ' '*(PatternQueueLen) # reset pattern buffer
            Buffer = '' # reset buffer
            if FirstSeq == 1: # need to skip over any header information
                FirstSeq = 0
                while CurrentLine == '': # until find a real line
                    CurrentLine = InputFile.readline().strip()
                # test for support file formats
                if CurrentLine[0:3].strip() == 'ID' or CurrentLine[0:6].strip() ==
'LOCUS':
                    # EMBL or GenBank formats, skip over headers
                    SequenceInfo = '>%s\n' % (CurrentLine)
                    while True:
                        CurrentLine = InputFile.readline()
                        if CurrentLine == '' or CurrentLine[0:6] == 'ORIGIN' or
CurrentLine[0:2] == 'SQ': break
                elif CurrentLine[0] == '>':
                    # FASTA format, ready to go now
                    SequenceInfo = '%s\n' % (CurrentLine)
```

```
                else:
                    # assume plain format, reset file
                    InputFile.close()
                    InputFile = open(InputFileName, 'r') # open the input file
        # Load Buffer
        NewLine = '' # holds new positions being read in
        while len(NewLine) < BufferSize: # until enough data is present
            CurrentInLine = InputFile.readline()
            if CurrentInLine == '': # end of file, search the positions normally "saved"
between loads to ensure search last few
                EOF = 1
                Adjustment = 1 - TargetLength
                PatternAdjustment = 0
                break
            else:
                CurrentLine = CurrentInLine.strip(StripChars) # remove whitespace &
numbers
                if CurrentLine != '': # i.e., isn't just white space
                    if CurrentLine[0] == '>' or CurrentLine[0:2] == '//': # end of
sequence marker
                        if CurrentLine[0:2] == '//': # end of EMBL or GenBank seq
                            # now read until reach next beginning or end of file
                            while CurrentLine[0:3].strip() != 'ID' and
CurrentLine[0:6].strip() != 'LOCUS' and CurrentLine != '':
                                CurrentLine = InputFile.readline()
                            NewSequenceInfo = '>%s\n' % (CurrentLine.strip())
                            while True:
                                CurrentLine = InputFile.readline() # remove whitespace
                                if CurrentLine == '' or CurrentLine[0:6] == 'ORIGIN' or
CurrentLine[0:2] == 'SQ': break
                            if CurrentLine == '':
                                EOF = 1
                            else:
                                BeginSeq = 1
                        else:
                            # otherwise just assume FASTA and therefore line is cleared
                            NewSequenceInfo = '%s\n' % (CurrentInLine.strip('\n')) # uses
CurrentInLine rather than current line to fix missing #s on end of line, if any
                            BeginSeq = 1
                        Adjustment = 1 - TargetLength # search end of sequence
                        PatternAdjustment = 0
                        break # to avoid loading any more into buffer
                    else: # extend buffer
                        CurrentLine = ''.join(CurrentLine.split()) # remove internal
whitespace
                        NewLine = '%s%s' % (NewLine, CurrentLine.upper())
        Buffer = '%s%s' % (Buffer[MinListLen:], NewLine) # update buffer, saving end
positions that were not previously searched
        CHQBuffer = '%s%s' % (CHQBuffer[MinQueueLen:MinListLen], Buffer) # same with
pattern buffer
        MaxCounter = MaxCounter + len(NewLine) # update master counter
        LowCounter = MaxCounter - len(Buffer) # update counter for presenting position #s
in target hits
        PatCounter = MaxCounter - len(CHQBuffer) # update counter for presenting position
#s in pattern hits
```

```
        # split outputtype = 2 from others here b/c quite different, so more efficient
(lines of code & speed) to do so
        if OutputType != 2:
            if len(Buffer) >= MinBufferLen: # if long enough to actually run
                for BufferPos in range(len(Buffer)+Adjustment): # for each position in
buffer to be examined on this run
                    Sequence = Buffer[BufferPos:BufferPos + TargetLength] # current
"window"
                    if Sequence in SeqMatches: # if this is a target sequence
                        TempQueue = [] # for storing data for patterns
                        for EachTarget in SeqMatches[Sequence]: # for each match for that
sequence
                            if EachTarget[1] < 0 and EachTarget[7] ==
Buffer[BufferPos+TargetLength:BufferPos+TargetLength+len(EachTarget[7])]: # first hit in
series, ensure any "extra" positions also match
                                BlankGaps = ['']*(-EachTarget[1]-1) # empty list with
enough space for gap values
                                HitList = [[BufferPos+TargetLength+len(EachTarget[7]),
EachTarget[3], EachTarget[4], EachTarget[2], BlankGaps, EachTarget[8]]] # initialize list
for tracking hits
                                for EachSeg in range(1, -EachTarget[1]): # for remaining
segments
                                    NewHitList = [] # for updating partial hits
                                    for EachHit in HitList: # for each partial hit so far
                                        for NextPos in range(EachHit[0] + EachHit[1],
EachHit[0] + EachHit[2]): # for all positions at right gap distance from the hit
                                            NewSequence =
Buffer[NextPos:NextPos+EachHit[5]] # determine sequence at this position
                                            if NewSequence in SeqMatches: # if new
sequence matches something
                                                for NextTarget in
SeqMatches[NewSequence]: # for each match at that position
                                                    if NextTarget[0] == EachTarget[0] and
NextTarget[1] == EachSeg: # if same series and next position
                                                        NewMismatches = EachHit[3] +
NextTarget[2] # increase mismatches for this hit
                                                        if NewMismatches <=
MaxMismatches: # if mismatch threshold is not exceeded, update hit information
                                                            NewHitList.append([NextPos+Ea
chHit[5], NextTarget[3], NextTarget[4], NewMismatches, EachHit[4], NextTarget[8]])
                                                        NewHitList[-1][4][EachSeg-1]
= '%s ' % (NextPos - EachHit[0])
                                    HitList = NewHitList # update master hit information
                                if HitList != []: # if a full hit was found
                                    if NumberPatterns > 0: # used to prevent slow
increase in memory usage that otherwise occurs
                                        for EachFullHit in HitList: # output & store info
for each
                                        # Target name has last character dropped to lose
"marker" for multiple sequences with same name
                                            if FirstHit[EachTarget[5]] == 1:
                                                FirstHit[EachTarget[5]] = 0
                                                OutputFile[EachTarget[5]].write(SequenceI
nfo)
                                            if EachTarget[6] == 0: # + strand
                                                OutputFile[EachTarget[5]].write('%s%s%s\t
%s%s\t%s\t%s\n' % (EachTarget[0][:-1]*OutputType, '\t'*OutputType,
LowCounter+BufferPos+1, Buffer[BufferPos:EachFullHit[0]], '\t+'*Mode, EachFullHit[3],
```

```
''.join(EachFullHit[4])))
                                                    TempQueue.append([Buffer[BufferPos:EachFu
llHit[0]], 1, EachTarget[0][:-1]])
                                                else: # - strand
                                                    OutputFile[EachTarget[5]].write('%s%s%s\t
%s\t-\t%s\t%s\n' % (EachTarget[0][:-1]*OutputType, '\t'*OutputType,
LowCounter+EachFullHit[0], ''.join([ComplementKeys[Char] for Char in
Buffer[BufferPos:EachFullHit[0]][::-1]]), EachFullHit[3], ''.join(EachFullHit[4][::-1])))
                                                    TempQueue.append([Buffer[BufferPos:EachFu
llHit[0]], -1, EachTarget[0][:-1]])
                                        else:
                                            for EachFullHit in HitList: # output & store info
for each
                                                if FirstHit[EachTarget[5]] == 1:
                                                    FirstHit[EachTarget[5]] = 0
                                                    OutputFile[EachTarget[5]].write(SequenceI
nfo)
                                                if EachTarget[6] == 0: # + strand
                                                    OutputFile[EachTarget[5]].write('%s%s%s\t
%s%s\t%s\t%s\n' % (EachTarget[0][:-1]*OutputType, '\t'*OutputType,
LowCounter+BufferPos+1, Buffer[BufferPos:EachFullHit[0]], '\t+'*Mode, EachFullHit[3],
''.join(EachFullHit[4])))
                                                else: # - strand
                                                    OutputFile[EachTarget[5]].write('%s%s%s\t
%s\t-\t%s\t%s\n' % (EachTarget[0][:-1]*OutputType, '\t'*OutputType,
LowCounter+EachFullHit[0], ''.join([ComplementKeys[Char] for Char in
Buffer[BufferPos:EachFullHit[0]][::-1]]), EachFullHit[3], ''.join(EachFullHit[4][::-1])))
                        if TempQueue != []: # if hits were found, remember their info
long term
                            CompleteHitQueue[LowCounter+BufferPos] = TempQueue
        else: # outputtype = 2
            if Mode == 1: # nucleic acid style output
                if OutputFirstBases == 1:
                    # write divider lines
                    OutputFile[0].write('\n%s' % (SequenceInfo[:-1]))
                    OutputFile[1].write('\n%s' % (SequenceInfo[:-1]))
                    # output first few bases to antisense
                    OutputFile[1].write(''.join(['\n%s\t%s' % ((LowCounter+BufferPos+1),
ComplementKeys[Buffer[BufferPos]]) for BufferPos in range(min(len(Buffer), MinBufferLen-
1))]))
                    OutputFirstBases = 0
                LastPos = 0
                if len(Buffer) >= MinBufferLen: # if long enough to actually run
                    for BufferPos in range(len(Buffer)+Adjustment): # for each position
in buffer to be examined on this run
                        Sequence = Buffer[BufferPos:BufferPos + TargetLength] # current
"window"
                        if Sequence in SeqMatches: # if this is a target sequence
                            if LastPos <= BufferPos: # flush bases w/o hit & shift into
appropriate frame
                                OutputFile[0].write(''.join(['\n%s\t%s' %
((LowCounter+CurPos+1), Buffer[CurPos]) for CurPos in range(LastPos, BufferPos+1)]))
                                OutputFile[1].write(''.join(['\n%s\t%s' %
((LowCounter+CurPos+TargetLength), ComplementKeys[Buffer[CurPos+TargetLength-1]]) for
CurPos in range(LastPos, BufferPos+1)]))
                                LastPos = BufferPos+1
                                OutputFile[0].write('\t'*((LowCounter+BufferPos)%TargetLe
ngth+1))
```

```
                                OutputFile[1].write('\t'*((LowCounter+BufferPos)%TargetLe
ngth+1))
                            if NumberPatterns > 0: # used to prevent slow memory
consumption
                                TempQueue = []
                                for EachTarget in SeqMatches[Sequence]: # for each match
at that address
                                    # output each target name (w/o last character)
matching this sequence & store in CHQ
                                    OutputFile[EachTarget[6]].write('%s ' %
(EachTarget[0][:-1]))
                                    if EachTarget[6] == 0: # + strand
                                        TempQueue.append([Buffer[BufferPos:BufferPos+Targ
etLength], 1, EachTarget[0][:-1]])
                                    else:
                                        TempQueue.append([Buffer[BufferPos:BufferPos+Targ
etLength], -1, EachTarget[0][:-1]])
                                CompleteHitQueue[LowCounter+BufferPos] = TempQueue
                            else:
                                for EachTarget in SeqMatches[Sequence]: # for each match
at that address
                                    # output each target name (w/o last character)
matching this sequence & store in CHQ
                                    OutputFile[EachTarget[6]].write('%s ' %
(EachTarget[0][:-1]))
                    # flush any remaining bases that did not have a hit
                    OutputFile[0].write(''.join(['\n%s\t%s' % ((LowCounter+CurPos+1),
Buffer[CurPos]) for CurPos in range(LastPos, BufferPos+1)]))
                    OutputFile[1].write(''.join(['\n%s\t%s' %
((LowCounter+CurPos+TargetLength), ComplementKeys[Buffer[CurPos+TargetLength-1]]) for
CurPos in range(LastPos, BufferPos+1)]))
            else: # protein style output
                LastPos = 0
                if len(Buffer) >= MinBufferLen: # if long enough to actually run
                    for BufferPos in range(len(Buffer)+Adjustment): # for each position
in buffer to be examined on this run
                        Sequence = Buffer[BufferPos:BufferPos + TargetLength] # current
"window"
                        if Sequence in SeqMatches: # if this is a target sequence
                            if LastPos <= BufferPos: # flush bases w/o hit & shift into
appropriate frame
                                OutputFile[0].write(''.join(['\n%s\t%s' %
((LowCounter+CurPos+1), Buffer[CurPos]) for CurPos in range(LastPos, BufferPos+1)]))
                                OutputFile[0].write('\t')
                                LastPos = BufferPos+1
                            if NumberPatterns > 0: # used to prevent slow memory
consumption
                                TempQueue = []
                                for EachTarget in SeqMatches[Sequence]: # for each match
at that address
                                    # output each target name (w/o last character)
matching this sequence & store in CHQ
                                    OutputFile[0].write('%s ' % (EachTarget[0][:-1]))
                                    TempQueue.append([Buffer[BufferPos:BufferPos+TargetLe
ngth], 1, EachTarget[0][:-1]])
                                CompleteHitQueue[LowCounter+BufferPos] = TempQueue
                            else:
                                for EachTarget in SeqMatches[Sequence]: # for each match
```

```
at that address
                                    # output each target name (w/o last character)
matching this sequence & store in CHQ
                                    OutputFile[EachTarget[6]].write('%s ' %
(EachTarget[0][:-1]))
                    # flush any remaining bases that did not have a hit
                    OutputFile[0].write(''.join(['\n%s\t%s' % ((LowCounter+CurPos+1),
Buffer[CurPos]) for CurPos in range(LastPos, BufferPos+1)]))

        # pattern matching
        if NumberPatterns > 0 and len(Buffer) >= MinBufferLen: # if patterns exist &
buffer is long enough to search
            KeyList=CompleteHitQueue.keys() # get position #s that have hits
            KeyList.sort() # sort position #s; often out of order otherwise, which will
cause pattern hits to be missed
            for HitPos in KeyList: # all positions with data in them
                if HitPos >= MaxCounter + PatternAdjustment: break # if saving for next
round
                PatNum = -1 # used for determining if should output sequence info
                for Pattern in PatternRestrictions: # for each pattern
                    PatNum = PatNum + 1
                    PatternHits = [] # for storing hits as they develop
                    for Firsts in CompleteHitQueue[HitPos]: # for each target sequence at
this position
                        # store name in proper place (with others of the same total
length so far)
                        for Partial in PatternHits:
                            if Partial[0] == HitPos + len(Firsts[0]) and Firsts[1] ==
Partial[3]: # if start of next position is same
                                Partial[1][0][Firsts[2]] = Firsts[2] # add name to list
at that distance
                                break
                        else:
                            PatternHits.append([HitPos + len(Firsts[0]),
[{Firsts[2]:Firsts[2]}], [Firsts[0]], Firsts[1]])
                    for Entry in Pattern[4][1:]: # for each pattern entry
                        NewPatternHits = [] # for updating PatternHits
                        if len(Entry) == 2: # if a gap
                            for Partial in PatternHits: # for each partial pattern so far
                                for NextPos in range(Partial[0]+Entry[0],
Partial[0]+Entry[1]+1):
                                    if NextPos in CompleteHitQueue: # if hits exist at
this position, create a new partial hit
                                        NewPatternHits.append([NextPos, Partial[1][:],
Partial[2][:], Partial[3]])
                                        NewPatternHits[-1][1].append([str(NextPos-
Partial[0])])
                                        NewPatternHits[-1][2].append('') # placeholder
                        else: # a "real" position
                            for Partial in PatternHits: # for each partial pattern so far
                                NewNewPatternHits = [] # for extending NewPatternHits
                                if Partial[0] in CompleteHitQueue: # if a hit here
                                    for EachHit in CompleteHitQueue[Partial[0]]: # each
hit at next position
                                        if Entry[1] != EachHit[1]*Partial[3] and
(Entry[0] == -1 or (Entry[0] > -1 and EachHit[2] in Partial[1][Entry[0]])): # if on
appropriate strand and name exists in previous position, if relevant
```

```
                                            # store name in proper place (with others of
the same total length so far)
                                            for NewPartial in NewNewPatternHits:
                                                if NewPartial[0] == Partial[0] +
len(EachHit[0]): # if start of next position is same
                                                    NewPartial[1][-1][EachHit[2]] =
EachHit[2] # add name to list at that distance
                                                    break
                                            else:
                                                if PatternType == 1 and Entry[0] > -1: #
if matching based on sequence, make sure they do or don't add this
                                                    if (Entry[1] == -1 and EachHit[0] !=
Partial[2][Entry[0]]) or (Entry[1] == 1 and EachHit[0] != ''.join([ComplementKeys[Char]
for Char in Partial[2][Entry[0]][::-1]])): break
                                                NewNewPatternHits.append([Partial[0] +
len(EachHit[0]), Partial[1][:], Partial[2][:], Partial[3]])
                                                NewNewPatternHits[-
1][1].append({EachHit[2]:EachHit[2]})
                                                NewNewPatternHits[-
1][2].append(EachHit[0])
                                NewPatternHits = NewPatternHits + NewNewPatternHits #
remember these new hits
                            for Partial in NewPatternHits: # update old positions for
cases where must be same as current one
                                Partial[1][Entry[0]] = Partial[1][-1] # self=self if no
positions need be the same
                        PatternHits = NewPatternHits # update master list
                        if PatternHits == []: break # stop cycling through pattern if no
partial hits remaining
                    for FullHit in PatternHits: # for each finished hit
                        # ensure unique positions actually are
                        if Pattern[0] == -1: # if any unique positions exist
                            for EntryNum in range(len(Pattern[4])): # length of pattern
                            if len(Pattern[4][EntryNum]) > 2 and
Pattern[4][EntryNum][2] == 1: # if a unique position
                                    # use OldPos & NewPos to avoid copy.deepcopy()
elsewhere when not really needed
                                    OldPos = FullHit[1][EntryNum].copy()
                                    for EachMatch in FullHit[1][EntryNum].keys(): # check
each entry for this position
                                        DelPos = 0 # to know if should delete this entry
at current position
                                        for OtherEntryNum in range(len(Pattern[4])): #
check all other positions
                                            if OtherEntryNum != EntryNum and
len(Pattern[4][OtherEntryNum]) > 2 and EachMatch in FullHit[1][OtherEntryNum]: # don't
check gaps or self
                                                CurPos = FullHit[1][OtherEntryNum].copy()
                                                DelPos = 1 # note need to remove from
position EntryNum
                                                del CurPos[EachMatch] # remove this entry
from this position
                                                FullHit[1][OtherEntryNum] = CurPos
                                        if DelPos == 1: # if matches were found with this
entry
                                            del OldPos[EachMatch] # delete this entry
                                    FullHit[1][EntryNum] = OldPos
                            NoOutput = 0
```

```
                            for EachPos in FullHit[1]: # make sure all positions in this
hit still have entries
                                if len(EachPos) == 0:
                                    NoOutput = 1
                                    break
                            if NoOutput == 1: continue # skip output if a position was
found to be empty now
                        # convert dictionaries to lists for output & determine longest
                        for EntryNum in range(len(Pattern[4])): # length of pattern
                            if len(Pattern[4][EntryNum]) > 2: # only do this on list of
hits, not gaps
                                FullHit[1][EntryNum] = FullHit[1][EntryNum].values()
                                FullHit[1][EntryNum].sort() # for consistency, otherwise
arrangement varies, but not required
                        if FirstPatHit[PatNum] == 1:
                            FirstPatHit[PatNum] = 0
                            PatternFile[Pattern[1]].write(SequenceInfo)
                        if FullHit[3] == 1: # + strand
                            PatternFile[Pattern[1]].write('%s\t%s%s%s' % (HitPos+1,
CHQBuffer[HitPos-PatCounter:FullHit[0]-PatCounter], '\t+'*Mode, ''.join(['\t' +
''.join([EachMatch + ' ' for EachMatch in EachPos]) for EachPos in FullHit[1]])))
                        else: # - strand
                            FullHit[1].reverse() # flip it to match headers in file
                            PatternFile[Pattern[1]].write('%s\t%s\t-%s' % (FullHit[0],
''.join([ComplementKeys[Char] for Char in CHQBuffer[HitPos-PatCounter:FullHit[0]-
PatCounter][::-1]]), ''.join(['\t' + ''.join([EachMatch + ' ' for EachMatch in EachPos])
for EachPos in FullHit[1]])))
                        PatternFile[Pattern[1]].write('\n')
                del CompleteHitQueue[HitPos] # finished with this position, so remove it
        if NewSequenceInfo != '': # copy new header into active header position
            SequenceInfo = NewSequenceInfo
            NewSequenceInfo = ''
    if OutputType == 2:
        # done, output remaining few bases/residues to sense strand/protein chain
        OutputFile[0].write(''.join(['\n%s\t%s' % ((LowCounter+BufferPos+1),
Buffer[BufferPos]) for BufferPos in range(max(len(Buffer)+MinListLen, 0), len(Buffer))]))
    # close all files
    InputFile.close()
    for File in OutputFile:
        File.close()
    for File in PatternFile.values():
        File.close()
#--------------------------------------------------------------------------
def GenerateMismatches(SplitTargets, Before, After, NumMis, Pos, MaxMis, MismatchKeys):
    """Recursively generates all possible mismatches for a given sequence by
        iterating through sequence 1 position at a time, keeping the info
        associated with the sequence the same as the master sequence.
    """
    if len(After) >= NumMis: # if not done with current cycle
        for CharPos in range(len(After)): # for each character not yet done
            if MismatchKeys[After[0]] != '': # For ones that do not have mismatches
(e.g., N & X)
                if NumMis == 1: # if current mismatch is done, store it
                    SplitTargets.append([Before + MismatchKeys[After[0]] + After[1:],
SplitTargets[Pos][1][:]]) # send a copy
                    SplitTargets[len(SplitTargets)-1][1][2] = MaxMis
```

```
                else: # move on to the next position to change to a mismatch
                    SplitTargets = GenerateMismatches(SplitTargets, Before +
MismatchKeys[After[0]], After[1:], NumMis-1, Pos, MaxMis, MismatchKeys)
            Before = Before + After[0] # shift to the sequence left to do
            After = After[1:]
    return SplitTargets
#-------------------------------------------------------------------------
def GetKeys(KeyFile):
    """Reads rules file for what each character encountered will mean.
    ComplementKeys determines what the complements to a base are.  This also
        sets Protein or Nucleic Acid Mode.  If anything is present here, then
        Nucleic Acid mode is assumed; if this area is blank, then Protein.
    MismatchKeys determines what letter means not the same as another letter.
    WildcardKeys determines what wildcard letters correspond to, as a list.
        To match wildcards in source sequence, must include themselves in list.
    MasterKeys is the same as ComplementKeys, with all other ASCII characters
        added as a precaution against crashes; they all act as 'undefined'
        positions if encountered in a source sequence.  It replaces
        ComplementKeys in the FindSequences routine.  For Protein Mode, this
        is just a list of all characters as a precaution against crashes.
    """
    TestFileExist(KeyFile) # see if file exists
    InputFile = open(KeyFile, 'r')
    WhichDic = 0 # for indicating which dictionary is being built
    Mode = 0
    ComplementKeys = {}
    MismatchKeys = {}
    WildcardKeys = {}
    for EachLine in InputFile:
        EachLine = EachLine.strip() # remove white space
        if EachLine.lower() == 'complements:':
            WhichDic = 1
        elif EachLine.lower() == 'mismatches:':
            WhichDic = 2
        elif EachLine.lower() == 'wildcards:':
            WhichDic = 3
        elif len(EachLine) > 0 and EachLine[0] != '#': # if not blank line
            Pairs = EachLine.split(',') # split into key/value pairs
            for EachEntry in Pairs:
                if EachEntry != '': # in case line break after comma
                    KeyVal = EachEntry.split('=') # split key/value apart
                    if WhichDic == 1:
                        ComplementKeys[KeyVal[0].strip()] = KeyVal[1].strip()
                    elif WhichDic == 2:
                        MismatchKeys[KeyVal[0].strip()] = KeyVal[1].strip()
                    elif WhichDic == 3:
                        WildcardKeys[KeyVal[0].strip()] = list(KeyVal[1].strip())
                    else:
                        AbortProgram('Error: rules file is missing header line.')
    # following segment adds all other characters to return themselves in case
encountered in source
    if ComplementKeys != {}:
        MasterKeys = ComplementKeys.copy()
```

```
        Mode = 1 # nucleic acid
    else:
        for EachKey in MismatchKeys.keys():
            ComplementKeys[EachKey] = EachKey
        Mode = 0
        MasterKeys = {}
    TempList = [chr(AsciiChar) for AsciiChar in range(0, 256)]
    for Char in TempList:
        if Char not in MasterKeys: # don't replace "real" keys
            MasterKeys[Char] = Char
    return Mode, ComplementKeys, MismatchKeys, WildcardKeys, MasterKeys
#---------------------------------------------------------------------------
def Initialization(IniFile):
    """Reads IniFile to determine search paramaters and options.
    Reads in all user settings, determines that essential information is there.
    If something critical is missing, it will exit the program.  All
    non-essential information is checked; if missing or invalid, a warning is
    issued and the default value is used.
    """
    TestFileExist(IniFile) # see if file exists
    InputFile = open(IniFile, 'r')
    Digits = '-0123456789'
    # initial parameters so can trap if they don't get set
    # ones that cause critical errors
    InputFiles = []
    DatabaseFile = ''
    KeyFile = ''
    # ones that cause warnings to be issued; if not set will get put to a default
    OutputDirectory = ' ' # default is current dir = ''
    InputDir = ' ' # default is current dir = ''
    PatternToMatch = ' ' # default is ''
    MaxMismatches = -1 # default is 0
    Mismatches = -1 # default is 0
    CreateDir = -1 # default is 0
    OutputType = -1 # default is 1
    StandardizeNames = -1 # default is 0
    PatternType = -1 # default is 0, meaning matches patterns by name
    Buffer = -1 # default is 1000
    RAMWarning = -1 # default is 50
    # parse ini file read into variables & search strings
    for EachLine in InputFile:
        ProcessArray = EachLine.split('=', 1) # get parameter & value
        CurIniEntry = ProcessArray[0].lower().strip() # remove white space & make
lowercase
        if len(ProcessArray) == 2: # if it had an = sign in entry
            CurIniValue = ProcessArray[1].strip()
            if CurIniEntry == 'pattern':
                PatternToMatch = CurIniValue
            elif CurIniEntry == 'database':
                DatabaseFile = CurIniValue
            elif CurIniEntry == 'source directory':
                InputDir = CurIniValue.replace('\\', '/')
                if len(InputDir) > 0 and InputDir[-1] != '/':
```

```
                    InputDir = InputDir + '/' # ensure ends with a final /
            elif CurIniEntry == 'output directory':
                OutputDirectory = CurIniValue.replace('\\', '/')
                if len(OutputDirectory) > 0 and OutputDirectory[-1] != '/':
                    OutputDirectory = OutputDirectory + '/' # ensure ends with a final /
            elif CurIniEntry == 'rules file':
                KeyFile = CurIniValue
            elif len(CurIniValue) > 0: # following assume non blank entry
                if CurIniEntry == 'source file':
                    InputFiles = [FileName.strip() for FileName in
ProcessArray[1].split(',')]
                elif CurIniValue.strip(Digits) == '': # following must contain only
numerals
                    if CurIniEntry == 'maximum mismatches per segment':
                        Mismatches = int(CurIniValue)
                    elif CurIniEntry == 'maximum total mismatches':
                        MaxMismatches = int(CurIniValue)
                    elif CurIniEntry == 'create directories':
                        CreateDir = int(CurIniValue)
                    elif CurIniEntry == 'output format':
                        OutputType = int(CurIniValue)
                    elif CurIniEntry == 'standardize name length':
                        StandardizeNames = int(CurIniValue)
                    elif CurIniEntry == 'exact sequence pattern match':
                        PatternType = int(CurIniValue)
                    elif CurIniEntry == 'buffer':
                        Buffer = int(CurIniValue)
                    elif CurIniEntry == 'ram warning level':
                        RAMWarning = int(CurIniValue)
            # ignore anything else
    InputFile.close()
    # error handling
    if InputFiles == []:
        AbortProgram('Error: "Source File=" line not found or is blank in initialization
file.')
    if DatabaseFile == '':
        AbortProgram('Error: "Database=" line not found or is blank in initialization
file.')
    else:
        # Get "name" of database, w/o extension or directory
        DatabaseFrag = DatabaseFile[:]
        for SlashPos in range(len(DatabaseFrag)-1, -1, -1):
            if DatabaseFrag[SlashPos] == '/':
                DatabaseFrag = DatabaseFrag[SlashPos+1:] # remove / and everything before
                break # because only looking for last one
        for DotPos in range(len(DatabaseFrag)-1, -1, -1):
            if DatabaseFrag[DotPos] == '.':
                DatabaseFrag = DatabaseFrag[:DotPos] + '_' # remove . and everything
after
                break # because only looking for last one
    if KeyFile == '':
        AbortProgram('Error: "Rules File=" line not found or is blank in initialization
file.')
    # "warning" handling: non-critical problems in .ini information
```

```
    if InputDir == ' ':
        print 'Warning: "Source Directory=" line not found in initialization file.'
        print '  Assuming current directory for source.'
        InputDir = ''
    if OutputDirectory == ' ':
        print 'Warning: "Output Directory=" line not found in initialization file.'
        print '  Assuming current directory for output.'
        OutputDirectory = ''
    if PatternToMatch == ' ':
        print 'Warning: "Pattern=" line not found in initialization file.'
        print '  Running program with no pattern.'
        PatternToMatch = ''
    elif PatternType != 1 and PatternType != 0:
        print 'Warning: "Exact Sequence Pattern Match=" line not found or has invalid
value (not 0 or 1) in initialization file.'
        print '  Running program with default value of 0.'
        PatternType = 0
    if MaxMismatches < 0:
        print 'Warning: "Maximum Total Mismatches=" line not found or has invalid value
in initialization file.'
        print '  Running program with no mismatches.'
        MaxMismatches = 0
    elif Mismatches < 0:
        print 'Warning: "Maximum Mismatches per Segment=" line not found or has invalid
value in initialization file.'
        print '  Running program with no mismatches.'
        Mismatches = 0
        MaxMismatches = 0
    elif MaxMismatches < Mismatches:
        print 'Warning: Maximum Total Mismatches < Mismatches per Segment.'
        print '  Increasing Maximum Total Mismatches to equal Mismatches per Segment.'
        MaxMismatches = Mismatches
    if CreateDir != 1 and CreateDir != 0:
        print 'Warning: "Create Directories=" line not found or has invalid value (not 0
or 1) in initialization file.'
        print '  Running program without directory creation.'
        CreateDir = 0
    if OutputType not in range(0, 3): # not 0, 1, or 2
        print 'Warning: "Output Format=" line not found or has invalid value (not 0, 1,
or 2) in initialization file.'
        print '  Running program with in default output mode (1).'
        OutputType = 1
    if StandardizeNames != 1 and StandardizeNames != 0:
        print 'Warning: "Standardize Name Length=" line not found or has invalid value
(not 0 or 1) in initialization file.'
        print '  Running program without standardizing lengths.'
        StandardizeNames = 0
    if Buffer <= 0:
        print 'Warning: "Buffer=" line not found or has invalid value (not a positive
integer) in initialization file.'
        print '  Running program with default buffer size of 1000.'
        Buffer = 1000
    if RAMWarning <= 0:
        print 'Warning: "RAM Warning Level=" line not found or has invalid value (not a
positive integer) in initialization file.'
```

```
        print '  Running program with default warning level of 50.'
        RAMWarning = 50
    # check for file names that are too similar (for directory creation) and/or identical
(would just be wasteful to run)
    # too similar is defined as identical source file names if extension and
capitalization is removed
    for FileNum in range(len(InputFiles)):
        LastDot1 = len(InputFiles[FileNum])
        for DotPos in range(LastDot1-1, -1, -1):
            if InputFiles[FileNum][DotPos] == '.': LastDot1 = DotPos
            break # found last . in file name
        for NextNum in range(FileNum+1, len(InputFiles)): # search all files after this
one
            LastDot2 = len(InputFiles[NextNum])
            for DotPos in range(LastDot2 - 1, -1, -1):
                if InputFiles[NextNum][DotPos] == '.': LastDot2 = DotPos
                break # found last .
            if InputFiles[FileNum][:LastDot1] == InputFiles[NextNum][:LastDot2]:
                AbortProgram('Error: identical or nearly identical (only extension
different) source files found.')


    # combine source directory with all source files
    InputFileName = [InputDir + FileName.strip() for FileName in InputFiles]
    return InputFileName, OutputDirectory, PatternToMatch, DatabaseFile, Mismatches,
MaxMismatches, CreateDir, OutputType, PatternType, DatabaseFrag, Buffer, RAMWarning,
KeyFile, StandardizeNames
#--------------------------------------------------------------------------
def LoadDatabase(DatabaseFile, Mismatches, OutputType, RAMThreshold, ComplementKeys,
MismatchKeys, WildcardKeys, Mode, DatabaseFrag, StandardizeNames):
    """Loads all target sequences to be searched for and parses them.
    Reads in all target sequences, breaks them up as appropriate,
    generates reverse complements, generates all mismatch sequences if needed,
    expands wildcards into multiple sequences, and stores result into
    SegMatches.  Also returns a variety of other relevant information for
    speeding things along later.
    The limiting factor in this routine is creation of SeqMatches; generally
    over 50% of the total processing time is the assignment statement
    (mostly generation of the hash value & memory creation).
    SeqMatches is dictionary with key = sequence, value = list of:
    [y] = all information about targets matching that sequence
    [y][0] = Names of target (with unique marker char appended)
    [y][1] = # this segment is in order (1=2nd, 2=3rd, etc.)
             if 1st position, then is -NumSegs for that sequence
    [y][2] = # of mismatches for this target at this sequence
    [y][3] = minimum gap length following this segment
    [y][4] = maximum gap length following this segment
    [y][5] = index in FileNameArray containing the relevant output file
    [y][6] = Strand (0 = sense, 1 = antisense)
    [y][7] = '' unless 1st in series & 1st segment is longer than TL,
                in which case it contains the extra positions
    [y][8] = Length of following segment
    """
    RAMThreshold = RAMThreshold*10000 # roughly 10K sequences per MB RAM
    TestFileExist(DatabaseFile) # see if database file exists
    InputFile = open(DatabaseFile, 'r')
```

```
    Digits = '0123456789' # negative values not allowed for gaps
    TargetList = [] # for storing targets
    TargetLength = 10**10
    MultSeqsForTarget = 0 # if target has multiple sequences associated with it, need to
do some extra error checking
    for EachLine in InputFile:
        ProcessArray = EachLine.split('=', 1) # split into names and sequences
        if len(ProcessArray) == 2: # only if this line contains an =
            MultArray = ProcessArray[1].split(',') # split sequences into separate parts
(if more than 1)
            for EachSeq in MultArray:
                TargetList.append([ProcessArray[0].strip(), EachSeq.strip().upper()]) #
remove white space, store name/seq pair
    InputFile.close()
    # ensure at least one target identified before proceeding
    if len(TargetList) == 0:
        AbortProgram('Error: no valid target sequences found in database.')
    if StandardizeNames == 1: # make all names same length
        LongestName = 0
        for EachEntry in TargetList: # determine longest name length
            if len(EachEntry[0]) > LongestName:
                LongestName = len(EachEntry[0]) # store longest name length
        for Pos in range(len(TargetList)): # adjust name lengths
            if len(TargetList[Pos][0]) < LongestName:
                TargetList[Pos][0] = TargetList[Pos][0] + ' '*(LongestName -
len(TargetList[Pos][0])) # append spaces to make all same length
    # generate list of names for files; all targets with same name must refer to same
file
    if OutputType == 0: # generate names based on
        FileNameArray = [] # stores file names
        for EachTarget in TargetList: # for each entry
            for PrevTarget in range(len(FileNameArray)): # for all previous file names
established
                if EachTarget[0].lower() == FileNameArray[PrevTarget].lower(): # if same
name
                    EachTarget.append(PrevTarget) # make this entry refer to this file
                    MultSeqsForTarget = 1 # and note that multiple sequences for single
target
                    break
            else: # need to create new file reference
                EachTarget.append(len(FileNameArray))
                FileNameArray.append(DatabaseFrag + EachTarget[0])
    else:
        # check for multiple entries for single target by removing duplicates & comparing
length
        TempDic = {}
        for EachTarget in TargetList:
            EachTarget.append(0)
            TempDic[EachTarget[0].lower()] = EachTarget[0]
        if len(TempDic) < len(TargetList):
            MultSeqsForTarget = 1
        if OutputType == 1: # single file output
            FileNameArray = [DatabaseFrag + 'output']
        else: # full annotated output
            if Mode == 1:
```

```
                FileNameArray = [DatabaseFrag + 'sense', DatabaseFrag + 'antisense']
            else:
                FileNameArray = [DatabaseFrag + 'sequence']
    # now append .txt to all file names
    for FileNum in range(len(FileNameArray)):
        FileNameArray[FileNum] = FileNameArray[FileNum] + '.txt'
    # now make each entry (even if same target name) unique to avoid "crossing over" of
multiple target sequences with gaps
    NameTracker = {}
    for TargetNum in range(len(TargetList)):
        if TargetList[TargetNum][0] in NameTracker:
            NameTracker[TargetList[TargetNum][0]] =
NameTracker[TargetList[TargetNum][0]]+2
            if NameTracker[TargetList[TargetNum][0]] > 128:
                AbortProgram('Error: too many listed sequences associated with a single
target name.')
        else:
            NameTracker[TargetList[TargetNum][0]] = 0
        TargetList[TargetNum][0] = TargetList[TargetNum][0] +
chr(127+NameTracker[TargetList[TargetNum][0]])
    SplitTargets = [] # for storing segments
    LongestTarget = 0
    ShortestTarget = 10**10 # presumably nobody will run a target longer than 10 billion
    for EachTarget in TargetList:
        # split each entry into segments
        SeqSegs = [] # sequence segments for this target
        GapSegs = [] # gap segments for this target
        # check for easy problems to catch
        if EachTarget[1][-1] == ']': # terminal gap
            AbortProgram('Error: terminal gap found in target sequence.')
        if EachTarget[1][0] == '[': # opening gap
            AbortProgram('Error: target sequence begins with gap.')
        if EachTarget[1].count('[') != EachTarget[1].count(']'): # unbalanced gap markers
            AbortProgram('Error: target sequence is missing opening or closing gap marker
([ or ]).')
        Portions = EachTarget[1].split('[') # Portions[0] = first sequence part, [x] =
[x-1 gap, x seq part]
        NumSegs = len(Portions) # Number of segments for this target
        # first segment does not have gap region attached, so just add it
        SeqSegs.append(Portions[0])
        # others need to have gap regions removed before adding to list
        for Remaining in range(1, len(Portions)):
            Portions[Remaining] = Portions[Remaining].split(']')
            SeqSegs.append(Portions[Remaining][1])
            GapSegs.append(Portions[Remaining][0].split(':'))
            if len(GapSegs[-1]) != 2: # if gap did not contain exactly 1 ":"
                AbortProgram('Error: target gaps must be in the format [X:Y], where X & Y
are non-negative integers.')
            else:
                for EachPart in GapSegs[-1]: # determine that all values are valid
                    if len(EachPart) == 0 or EachPart.strip(Digits) != '':
                        AbortProgram('Error: target gaps must be in the format [X:Y],
where X & Y are non-negative integers.')
        # SeqSegs is now list of all segments, GapSegs is list of [first, second] of gap
elements
```

```
        for EachSeg in SeqSegs:
            if len(EachSeg) == 0: # if 0 length segment
                AbortProgram('Error: target contains 2 gaps without sequence between
them.')
            else: # ensure all characters exist in dictionaries
                for Char in EachSeg:
                    if Char not in ComplementKeys:
                        AbortProgram('Error: target sequence contains a character (' +
str(Char) + ') not in complement rule set.')
                    elif Char not in MismatchKeys:
                        AbortProgram('Error: target sequence contains a character (' +
str(Char) + ') not in mismatch rule set.')
        # determine low & high values for each gap region, maximum value
        MinRegion = []
        MaxRegion = []
        for GapNum in range(len(GapSegs)): # for each gap segment, determine max and min,
determine longest overall
            MinRegion.append(min(int(GapSegs[GapNum][0]), int(GapSegs[GapNum][1])))
            MaxRegion.append(max(int(GapSegs[GapNum][0]), int(GapSegs[GapNum][1]))+1) #
+1 to include this value when calculating ranges
        # append all information for each segment
        for SegNum in range(NumSegs-1):
            SplitTargets.append([SeqSegs[SegNum], [EachTarget[0], SegNum, 0,
MinRegion[SegNum], MaxRegion[SegNum], EachTarget[2], 0, '', len(SeqSegs[SegNum+1])]])
        SplitTargets.append([SeqSegs[NumSegs-1], [EachTarget[0], NumSegs-1, 0, 0, 0,
EachTarget[2], 0, '', 0]])
        SplitTargets[-NumSegs][1][1] = -NumSegs
        # generate reverse complements to each segment
        if Mode == 1: # nucleic acid style
            comparray = []
            for SegNum in range(NumSegs): # for each sequence Segment, get rc & append
                compstr = ''.join([ComplementKeys[Char] for Char in SeqSegs[SegNum][::-
1]])
                comparray.append(compstr)
            comparray.reverse()
            MaxRegion.reverse()
            MinRegion.reverse()
            RCName = EachTarget[0][:-1] + chr(ord(EachTarget[0][-1])+1)
            for SegNum in range(NumSegs-1):
                SplitTargets.append([comparray[SegNum], [RCName, SegNum, 0,
MinRegion[SegNum], MaxRegion[SegNum], EachTarget[2], 1, '', len(comparray[SegNum+1])]])
            SplitTargets.append([comparray[NumSegs-1], [RCName, NumSegs-1, 0, 0, 0,
EachTarget[2], 1, '', 0]])
            SplitTargets[-NumSegs][1][1] = -NumSegs
            TargetLength = min(TargetLength, len(SeqSegs[0]), len(SeqSegs[-1]))
        else:
            TargetLength = min(TargetLength, len(SeqSegs[0]))
        # determine longest & shortest possible Target
        CurMinLength = sum([len(EachSeg) for EachSeg in SeqSegs]) + sum(MinRegion)
        CurMaxLength = sum([len(EachSeg) for EachSeg in SeqSegs]) + sum(MaxRegion)
        LongestTarget = max(LongestTarget, CurMaxLength)
        ShortestTarget = min(ShortestTarget, CurMinLength)
        # if user requested output format 2 with variable target size, switch output type
        if OutputType == 2 and (ShortestTarget != LongestTarget or NumSegs > 1):
            print 'Warning: output format 2 only works when all target sequences are the
```

```
same length and do not contain gaps.'
            print '  Changing output format to default (type 1).'
            FileNameArray = [DatabaseFrag + 'output.txt']
            OutputType = 1
    # generate all possible mismatches
    splitlen = len(SplitTargets)
    for MismatchNum in range(1, Mismatches+1):
        for TargetNum in range(splitlen):
            SplitTargets = GenerateMismatches(SplitTargets, '',
SplitTargets[TargetNum][0], MismatchNum, TargetNum, MismatchNum, MismatchKeys)
    # now calculate how many sequences are going to be generated & ensure all chars are
in wildcards dictionary
    Hits = 0
    for EachTarget in SplitTargets:
        TempHits = 1
        for Char in EachTarget[0]:
            if Char in WildcardKeys:
                TempHits = TempHits*len(WildcardKeys[Char])
            else:
                AbortProgram('Error: generated sequence contains a character (' +
str(Char) + ') not in wildcard rule set.')
        Hits = Hits + TempHits
    if Hits > RAMThreshold: # if RAM usage is likely to be excessive
        ExcessFactor = str(int(100.0*(Hits-RAMThreshold)/RAMThreshold)) + '%'
        MessageStr = 'Warning: estimated RAM usage is ' + ExcessFactor + ' higher than
warning level.  Continue? '
        InputStr = ''
        while InputStr != 'y' and InputStr != 'n':
            InputFromUser = raw_input(MessageStr)
            InputStr = InputFromUser[0:1].lower()
        if InputStr == 'n':
            AbortProgram('Exiting.')
    # now need to expand all forms into single matches & store in master array
    SeqMatches = {}
    for EachTarget in SplitTargets: # for each segment
        TempList = [''] # reset
        for Char in EachTarget[0]: # for each character in segment sequences
            curlen = len(WildcardKeys[Char]) # number of characters corresponding to that
letter
            # duplicate list to allow for new entries
            TempList = TempList * curlen
            for NewCharNum in range(curlen): # for each conversion necessary, run through
a fraction of the list and replace the letter
                ListStart = NewCharNum * len(TempList)/curlen
                ListStop = (NewCharNum + 1) * len(TempList)/curlen
                for ListPos in range(ListStart, ListStop):
                    TempList[ListPos] = TempList[ListPos] +
WildcardKeys[Char][NewCharNum]
        # store all permutations from a SplitTargets entry in SeqMatches
        if EachTarget[1][1] < 0 and len(EachTarget[0]) > TargetLength:
            NeedCopy = 1 # all 1st sequences must be represented by consistent length, so
allow for moving excess into data array
        else:
            NeedCopy = 0
            TargetData = EachTarget[1]
```

```
        for NewTarget in TempList:
            if NeedCopy == 1: # duplicate to allow storage of excess sequence info
                TargetStr = NewTarget[:TargetLength]
                TargetData = EachTarget[1][:]
                TargetData[7] = NewTarget[TargetLength:]
            else:
                TargetStr = NewTarget
            if TargetStr in SeqMatches:
                PrevMatch = SeqMatches[TargetStr]
                if MultSeqsForTarget == 1: # possibility for single target/sequence pair
to generate same sequence w/ different mismatch levels, so find lowest
                    for MatchNum in range(len(PrevMatch)):
                        # if same name/series/strand, same position, and same extension
                        if PrevMatch[MatchNum][0] == TargetData[0] and
PrevMatch[MatchNum][7] == TargetData[7] and PrevMatch[MatchNum][1] == TargetData[1]:
                            if PrevMatch[MatchNum][2] > TargetData[2]: # replace existing
if newer has fewer mismatches
                                PrevMatch[MatchNum] = TargetData
                            # else don't need to add this one, skip it
                            break
                    else:
                        PrevMatch.append(TargetData) # otherwise just add it as a new
match for this sequence
                else:
                    PrevMatch.append(TargetData) # No need to worry about mismatch levels
            else:
                SeqMatches[TargetStr] = [TargetData] # create entry
    return TargetLength, SeqMatches, FileNameArray, OutputType, ShortestTarget,
LongestTarget
#-------------------------------------------------------------------------
def TestFileExist(FileName):
    """Determines if FileName exists, aborts script if not found.
    """
    try:
        TestFile = open(FileName, 'r')
    except IOError:
        AbortProgram('Error: ' + FileName +  ' not found.\n')
    TestFile.close()
#-------------------------------------------------------------------------
def TestFileOverWrite(OutputDirs, OutputType, FileNameArray, PatternRestrictions):
    """Determines if FileName exists, and confirms overwrite if so.
    """
    for OutDir in OutputDirs:
        FileList = []
        for FileName in FileNameArray:
            FileList.append(FileName)
        for PatNum in range(len(PatternRestrictions)):
            FileList.append(PatternRestrictions[PatNum][2])
        for FileName in FileList: # For each file to be created in this directory
            TestFileName = OutDir + FileName
            try:
                TestFile = open(TestFileName, 'r')
            except IOError:
                pass # file does not exist, so do nothing
```

```
            else: # does exist, so confirm overwrite
                MessageStr = 'Do you wish to overwrite existing files? '
                InputStr = ''
                while InputStr != 'y' and InputStr != 'n':
                    InputFromUser = raw_input(MessageStr)
                    InputStr = InputFromUser[0:1].lower()
                if InputStr == 'n':
                    AbortProgram('Exiting: provide a new output directory or move
exisiting files to avoid overwrites.')
                TestFile.close()
                return # so only prompts user once even if more files exist
#--------------------------------------------------------------------------
# Main program, mostly just calls routines.
# Routines allow psyco to accelerate code if available & ease finding of code
try: # try to load psyco, expecting that it may not be installed
    import psyco
except:
    pass # just don't load it
else:
    psyco.full()
SettingsFile = 'espsearch.ini' # the basic user settings file
print 'Initializing...'
# get initial variables from user
InputFileName, OutputDirectory, PatternToMatch, DatabaseFile, Mismatches, MaxMismatches,
CreateDir, OutputType, PatternType, DatabaseFrag, BufferSize, IssueRAMWarning, KeyFile,
StandardizeNames = Initialization(SettingsFile)
# get rules from user
Mode, ComplementKeys, MismatchKeys, WildcardKeys, MasterKeys = GetKeys(KeyFile)
# load target sequence database and parse it
TargetLength, SeqMatches, FileNameArray, OutputType, ShortestTarget, LongestTarget =
LoadDatabase(DatabaseFile, Mismatches, OutputType, IssueRAMWarning, ComplementKeys,
MismatchKeys, WildcardKeys, Mode, DatabaseFrag, StandardizeNames)
# parse patterns specified by user
PatternRestrictions, MinPatternBuffer = DefinePattern(PatternToMatch, ShortestTarget,
LongestTarget, Mode, DatabaseFrag)
# create directories if necessary, test for existence of source file(s)
if CreateDir == 0:
    # avoid creating directories, so no looping over multiple source files
    if len(InputFileName) > 1:
        print 'Warning: not creating directories, so will only process first source
file.'
    OutputDirs = [OutputDirectory]
    InputFileName = [InputFileName[0]]
    # test to ensure that the input file actually exists; if not, then exit
    TestFileExist(InputFileName[0])
else:
    # test to ensure that the input file(s) actually exist; if not, then exit
    for FileName in InputFileName:
        TestFileExist(FileName)
    # Tests for ability to write files and creates directories
    OutputDirs = DoOSOperations(InputFileName, OutputDirectory)
print 'Done.'
# test for overwrites
TestFileOverWrite(OutputDirs, OutputType, FileNameArray, PatternRestrictions)
# run through all source files
```

```
for FileNum in range(len(InputFileName)):
    print '  Searching', InputFileName[FileNum]
    FindSequences(SeqMatches, InputFileName[FileNum], OutputDirs[FileNum], TargetLength,
MaxMismatches, FileNameArray, PatternRestrictions, OutputType, PatternType, BufferSize,
ShortestTarget, LongestTarget, MinPatternBuffer, MasterKeys, Mode)
AbortProgram('Analysis complete.')
```

APPENDIX B
ESPSEARCHGUI CODE

```
# ESPSearchGUI.py, version 1.01
""" Graphical user interface for ESPSearch.
    Copyright (C) 2005 Terry J. Watt
    This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
    You may contact Terry Watt at:
    terry.watt@chemistry.gatech.edu
    School of Chemistry & Biochemistry
    Atlanta, GA 30332-0400
    See espsearchgui.html for instructions on using this script and for the GPL.
    See http://web.chemistry.gatech.edu/~doyle/espsearch/ for more information.
"""
#---------------------------------------------------------------------------
def ChooseDir(Key, Fields):
    """Opens dialog to choose directory & sets the corresponding field.
    """
    Diag = tkFileDialog
    dirname = Diag.askdirectory()
    if dirname != None:
        Fields[Key].delete(0, Tkinter.END)
        Fields[Key].insert(0, dirname)
#---------------------------------------------------------------------------
def ChooseFile(Key, Fields):
    """Opens dialog to choose file & sets the corresponding field.
    """
    Diag = tkFileDialog
    filename = Diag.askopenfile(title='Select File')
    if filename != None:
        filename.close()
        InsertText = str(filename).split('\'')
        Fields[Key].delete(0, Tkinter.END)
        Fields[Key].insert(0, InsertText[1])
#---------------------------------------------------------------------------
def ChooseFiles(Key, DirKey, Fields):
    """Opens dialog to choose multiple files, and sets list & directory separately.
    """
    Diag = tkFileDialog
    filenames = Diag.askopenfiles(title='Select File(s)')
    if filenames != '':
```

```
        InsertText = ''
        for EachName in filenames:
            EachName.close()
            Name = str(EachName).split('\'')
            RealName = Name[1].split('/')
            InsertText = InsertText + ', ' + RealName[-1]
        Fields[Key].delete(0, Tkinter.END)
        Fields[Key].insert(0, InsertText[2:])
        DirText = Name[1][:-len(RealName[-1])]
        Fields[DirKey].config(state=Tkinter.NORMAL)
        Fields[DirKey].delete(0, Tkinter.END)
        Fields[DirKey].insert(0, DirText)
        Fields[DirKey].config(state=Tkinter.DISABLED)
#------------------------------------------------------------------------------
def DrawEverything(root, SettingsFile):
    """Draws GUI, returning widgets that need to be manipulated.
    Note: command=Func(1,2,3,...) triggers call, and links command to the return value
        instead use: command=lambda A=1, B=2, C=3, ... : Func(A,B,C,...)
    """
    # Parameter labels
    TitleFont = ('Helvetica', 9, 'bold')
    MainFont = ('Helvetica', 8)
    HiddenFont = ('Helvetica', 4)
    LabelA = Tkinter.Label(text='File Locations', font=TitleFont)
    LabelA.grid(row=0, column=0, columnspan=5)
    Label1 = Tkinter.Label(text='Database', font=MainFont)
    Label1.grid(row=1, column=0, sticky=Tkinter.E)
    Label2 = Tkinter.Label(text='Output Directory', font=MainFont)
    Label2.grid(row=2, column=0, sticky=Tkinter.E)
    Label3 = Tkinter.Label(text='Rules File', font=MainFont)
    Label3.grid(row=3, column=0, sticky=Tkinter.E)
    Label4 = Tkinter.Label(text='Source Directory', font=MainFont)
    Label4.grid(row=4, column=0, sticky=Tkinter.E)
    Label5 = Tkinter.Label(text='Source File(s)', font=MainFont)
    Label5.grid(row=5, column=0, sticky=Tkinter.E)
    LabelAA = Tkinter.Label(text='', font=HiddenFont)
    LabelAA.grid(row=6, column=0, columnspan=5)
    LabelB = Tkinter.Label(text='General Options', font=TitleFont)
    LabelB.grid(row=10, column=0, columnspan=5)
    Label6 = Tkinter.Label(text='Create Directories', font=MainFont)
    Label6.grid(row=11, column=0, sticky=Tkinter.E)
    Label7 = Tkinter.Label(text='Output Format', font=MainFont)
    Label7.grid(row=12, column=0, sticky=Tkinter.E)
    Label8 = Tkinter.Label(text='Standarize Name Length', font=MainFont)
    Label8.grid(row=13, column=0, sticky=Tkinter.E)
    LabelBB = Tkinter.Label(text='', font=HiddenFont)
    LabelBB.grid(row=14, column=0, columnspan=5)
    LabelC = Tkinter.Label(text='Target Sequence Options', font=TitleFont)
    LabelC.grid(row=20, column=0, columnspan=5)
    Label9 = Tkinter.Label(text='Maximum Mismatches per Segment', font=MainFont)
    Label9.grid(row=21, column=0, sticky=Tkinter.E)
    Label10 = Tkinter.Label(text='Maximum Total Mismatches', font=MainFont)
    Label10.grid(row=22, column=0, sticky=Tkinter.E)
```

```
    LabelCC = Tkinter.Label(text='', font=HiddenFont)
    LabelCC.grid(row=23, column=0, columnspan=5)
    LabelD = Tkinter.Label(text='Pattern Options', font=TitleFont)
    LabelD.grid(row=30, column=0, columnspan=5)
    Label11 = Tkinter.Label(text='Exact Sequence Pattern Match', font=MainFont)
    Label11.grid(row=31, column=0, sticky=Tkinter.E)
    Label12 = Tkinter.Label(text='Pattern(s)', font=MainFont)
    Label12.grid(row=32, column=0, sticky=Tkinter.E)
    LabelDD = Tkinter.Label(text='', font=HiddenFont)
    LabelDD.grid(row=33, column=0, columnspan=5)
    LabelE = Tkinter.Label(text='Performance Options', font=TitleFont)
    LabelE.grid(row=40, column=0, columnspan=5)
    Label13 = Tkinter.Label(text='Buffer', font=MainFont)
    Label13.grid(row=41, column=0, sticky=Tkinter.E)
    Label14 = Tkinter.Label(text='RAM Warning Level', font=MainFont)
    Label14.grid(row=42, column=0, sticky=Tkinter.E)
    LabelEE = Tkinter.Label(text='', font=HiddenFont)
    LabelEE.grid(row=43, column=0, columnspan=5)
    # Entry fields
    Fields = {}
    Fields['Database'] = Tkinter.Entry(width=40)
    Fields['Database'].grid(row=1, column=1, padx=5, columnspan=3)
    DBButton = Tkinter.Button(text="Choose...", command=lambda A = 'Database', B =
Fields: ChooseFile(A, B))
    DBButton.grid(row=1, column=4)
    Fields['Output Directory'] = Tkinter.Entry(width=40)
    Fields['Output Directory'].grid(row=2, column=1, padx=5, columnspan=3)
    ODButton = Tkinter.Button(text="Choose...", command=lambda A = 'Output Directory', B
= Fields: ChooseDir(A, B))
    ODButton.grid(row=2, column=4)
    Fields['Rules File'] = Tkinter.Entry(width=40)
    Fields['Rules File'].grid(row=3, column=1, padx=5, columnspan=3)
    RFButton = Tkinter.Button(text="Choose...", command=lambda A = 'Rules File', B =
Fields: ChooseFile(A, B))
    RFButton.grid(row=3, column=4)
    Fields['Source Directory'] = Tkinter.Entry(width=40, state=Tkinter.DISABLED)
    Fields['Source Directory'].grid(row=4, column=1, padx=5, columnspan=3)
    Fields['Source File'] = Tkinter.Entry(width=40)
    Fields['Source File'].grid(row=5, column=1, padx=5, columnspan=3)
    SDButton = Tkinter.Button(text="Choose...", command=lambda A = 'Source File', B =
'Source Directory', C = Fields: ChooseFiles(A, B, C))
    SDButton.grid(row=5, column=4)
    Fields['Create Directories Var'] = Tkinter.IntVar()
    Fields['Create Directories'] = Tkinter.Checkbutton(variable=Fields['Create
Directories Var'])
    Fields['Create Directories'].grid(row=11, column=1, padx=5, sticky=Tkinter.W)
    Fields['Output Format Var'] = Tkinter.IntVar()
    Fields['Output Format 1'] = Tkinter.Radiobutton(text='Many Files',
variable=Fields['Output Format Var'], value=0, anchor=Tkinter.W)
    Fields['Output Format 1'].grid(row=12, column=1, padx=5, sticky=Tkinter.W)
    Fields['Output Format 2'] = Tkinter.Radiobutton(text='Single File',
variable=Fields['Output Format Var'], value=1, anchor=Tkinter.W)
    Fields['Output Format 2'].grid(row=12, column=2, padx=5, sticky=Tkinter.W)
    Fields['Output Format 3'] = Tkinter.Radiobutton(text='Strands',
variable=Fields['Output Format Var'], value=2, anchor=Tkinter.W)
```

```
    Fields['Output Format 3'].grid(row=12, column=3, padx=5, sticky=Tkinter.W)
    Fields['Output Format Var'].set(1)
    Fields['Name Length Var'] = Tkinter.IntVar()
    Fields['Standardize Name Length'] = Tkinter.Checkbutton(variable=Fields['Name Length
Var'])
    Fields['Standardize Name Length'].grid(row=13, column=1, padx=5, sticky=Tkinter.W)
    Fields['Maximum Mismatches per Segment'] = Tkinter.Entry(state=Tkinter.NORMAL,
width=5)
    Fields['Maximum Mismatches per Segment'].grid(row=21, column=1, padx=5,
sticky=Tkinter.W)
    Fields['Maximum Mismatches per Segment'].insert(0, '0')
    Fields['Maximum Total Mismatches'] = Tkinter.Entry(state=Tkinter.NORMAL, width=5)
    Fields['Maximum Total Mismatches'].grid(row=22, column=1, padx=5, sticky=Tkinter.W)
    Fields['Maximum Total Mismatches'].insert(0, '0')
    Fields['Pattern Var'] = Tkinter.IntVar()
    Fields['Exact Sequence Pattern Match'] = Tkinter.Checkbutton(variable=Fields['Pattern
Var'])
    Fields['Exact Sequence Pattern Match'].grid(row=31, column=1, padx=5,
sticky=Tkinter.W)
    Fields['Pattern'] = Tkinter.Entry(state=Tkinter.NORMAL, width=40)
    Fields['Pattern'].grid(row=32, column=1, padx=5, columnspan=3)
    Fields['Buffer'] = Tkinter.Entry(state=Tkinter.NORMAL, width=5)
    Fields['Buffer'].grid(row=41, column=1, padx=5, sticky=Tkinter.W)
    Fields['Buffer'].insert(0, '1000')
    Fields['RAM Warning Level'] = Tkinter.Entry(state=Tkinter.NORMAL, width=5)
    Fields['RAM Warning Level'].grid(row=42, column=1, padx=5, sticky=Tkinter.W)
    Fields['RAM Warning Level'].insert(0, '50')
    # command buttons
    FrameA = Tkinter.Frame(borderwidth=2, relief=Tkinter.RIDGE)
    FrameA.grid(row=50, column=0, columnspan=5)
    LoadButton = Tkinter.Button(FrameA, text='Load Settings', height=2, width=15,
command=lambda A = Fields : LoadSettings(A))
    LoadButton.grid(row=50, column=0, padx=10, pady=5)
    SaveButton = Tkinter.Button(FrameA, text='Save Settings', height=2, width=15,
command=lambda A = Fields : SaveSettings(A))
    SaveButton.grid(row=50, column=1, padx=10, pady=5)
    RunButton = Tkinter.Button(FrameA, text='Run Search', height=2, width=15,
command=lambda A = SettingsFile, B = Fields, C = root : RunSearch(A, B, C))
    RunButton.grid(row=50, column=2, padx=10, pady=5)
    QuitButton = Tkinter.Button(FrameA, text='Quit', height=2, width=15,
command=root.quit)
    QuitButton.grid(row=50, column=3, padx=10, pady=5)
    return Fields
#---------------------------------------------------------------------------
def LoadSettings(Fields):
    """Loads ini file & dumps into appropriate fields.
    No error checking is performed, ESPSearch handles all.
    For check/radio button values, unrecognized values are treated as default.
    For all, missing values are treated as default.
    """
    # prompt user for file name
    Diag = tkFileDialog
    InputFile = Diag.askopenfile(filetypes = [('Settings Files', '*.ini'), ('All Files',
'*')])
    # don't try to load if they didn't select a file
```

```
    if InputFile == None: return
    # clear/reset all fields to defaults
    Fields['Database'].delete(0, Tkinter.END)
    Fields['Output Directory'].delete(0, Tkinter.END)
    Fields['Rules File'].delete(0, Tkinter.END)
    Fields['Source Directory'].config(state=Tkinter.NORMAL) # active for editing
    Fields['Source Directory'].delete(0, Tkinter.END)
    Fields['Source File'].delete(0, Tkinter.END)
    Fields['Create Directories Var'].set(0)
    Fields['Output Format Var'].set(1)
    Fields['Name Length Var'].set(0)
    Fields['Maximum Mismatches per Segment'].delete(0, Tkinter.END)
    Fields['Maximum Mismatches per Segment'].insert(0, '0')
    Fields['Maximum Total Mismatches'].delete(0, Tkinter.END)
    Fields['Maximum Total Mismatches'].insert(0, '0')
    Fields['Pattern Var'].set(0)
    Fields['Pattern'].delete(0, Tkinter.END)
    Fields['Buffer'].delete(0, Tkinter.END)
    Fields['Buffer'].insert(0, '1000')
    Fields['RAM Warning Level'].delete(0, Tkinter.END)
    Fields['RAM Warning Level'].insert(0, '50')
    # load parameters & set fields
    for EachLine in InputFile:
        ProcessArray = EachLine.split('=', 1) # get parameter & value
        CurIniEntry = ProcessArray[0].lower().strip() # remove white space & make
lowercase
        if len(ProcessArray) == 2: # if it had an = sign in entry
            CurIniValue = ProcessArray[1].strip()
            if CurIniEntry == 'pattern':
                Fields['Pattern'].insert(0, CurIniValue)
            elif CurIniEntry == 'database':
                Fields['Database'].insert(0, CurIniValue)
            elif CurIniEntry == 'source directory':
                Fields['Source Directory'].insert(0, CurIniValue)
            elif CurIniEntry == 'output directory':
                Fields['Output Directory'].insert(0, CurIniValue)
            elif CurIniEntry == 'rules file':
                Fields['Rules File'].insert(0, CurIniValue)
            elif CurIniEntry == 'source file':
                Fields['Source File'].insert(0, CurIniValue)
            elif CurIniEntry == 'maximum mismatches per segment':
                Fields['Maximum Mismatches per Segment'].delete(0, Tkinter.END)
                Fields['Maximum Mismatches per Segment'].insert(0, CurIniValue)
            elif CurIniEntry == 'maximum total mismatches':
                Fields['Maximum Total Mismatches'].delete(0, Tkinter.END)
                Fields['Maximum Total Mismatches'].insert(0, CurIniValue)
            elif CurIniEntry == 'buffer':
                Fields['Buffer'].delete(0, Tkinter.END)
                Fields['Buffer'].insert(0, CurIniValue)
            elif CurIniEntry == 'ram warning level':
                Fields['RAM Warning Level'].delete(0, Tkinter.END)
                Fields['RAM Warning Level'].insert(0, CurIniValue)
            # next four actually require some processing
```

```
            elif CurIniEntry == 'create directories':
                if CurIniValue == '1':
                    Fields['Create Directories Var'].set(1)
            elif CurIniEntry == 'output format':
                if CurIniValue == '0':
                    Fields['Output Format Var'].set(0)
                elif CurIniValue == '2':
                    Fields['Output Format Var'].set(2)
            elif CurIniEntry == 'standardize name length':
                if CurIniValue == '1':
                    Fields['Name Length Var'].set(1)
            elif CurIniEntry == 'exact sequence pattern match':
                if CurIniValue == '1':
                    Fields['Pattern Var'].set(1)
            # ignore anything else
    InputFile.close()
    Fields['Source Directory'].config(state=Tkinter.DISABLED) # turn this off again
#--------------------------------------------------------------------------
def RunSearch(SettingsFile, Fields, root):
    """Runs main search, locking access to gui in meantime.
    Using redefined stdin & stdout, traps ESPSearch output & displays in a log
    window, and prompts the user for input when needed.  The log window will
    close (and allow access to main window) when user closes, following
    ESPSearch completion (whether successful or not).
    """
    SaveSettings(Fields, SettingsFile)
    class StdoutRedirect: # writes to a text box instead of Python display
        def __init__(self):
            self.last = '' # used by prompts in StdinRedirect
        def write(self, stuff):
            StatusDisplay.config(state=Tkinter.NORMAL)
            StatusDisplay.insert(Tkinter.END, stuff)
            StatusDisplay.yview(Tkinter.END)
            StatusDisplay.config(state=Tkinter.DISABLED)
            self.last = stuff
    class StdinRedirect: # pops up an input dialog box whenever readline called
(raw_input by ESPSearch)
        def readline(self):
            Prompt = StdoutDisplay.last
            Response = tkSimpleDialog.askstring('ESPSearch requires input', Prompt,
parent=DialogWindow)
            Response = Response + '\n'
            print '\n', Response[:-1]
            return Response

    def CloseWindow(): # used to return control to main window
        if ButtonState == Tkinter.NORMAL: # only if ESPSearch has finished
            DialogWindow.withdraw() # release control
            DialogWindow.update_idletasks() # update background
            root.focus_set() # return focus to main window
            DialogWindow.destroy() # remove this window

    # create dialog box
```

```
    DialogWindow = Tkinter.Toplevel() # define new main window
    DialogWindow.title('ESPSearch Status')
    DialogWindow.transient(root) # make not appear in taskbar & link max/min to main
window
    DialogWindow.grab_set() # make modal
    DialogWindow.initial_focus = DialogWindow # set focus to this window
    DialogWindow.protocol('WM_DELETE_WINDOW', lambda : CloseWindow()) # tie to OK button
    DialogWindow.geometry('+%d+%d' % (root.winfo_rootx()+50, root.winfo_rooty()+50)) #
position window 50 right & down from main window (upper right corners)

    # create a log for ESPSearch output & associate a scrollbar
    StatusDisplayScrollbar = Tkinter.Scrollbar(master=DialogWindow)
    StatusDisplayScrollbar.grid(row=0, column=1,
sticky=Tkinter.W+Tkinter.N+Tkinter.S+Tkinter.E)
    StatusDisplay = Tkinter.Text(master=DialogWindow, height=10, state=Tkinter.DISABLED,
yscrollcommand=StatusDisplayScrollbar.set, wrap=Tkinter.WORD, width=80)
    StatusDisplay.grid(row=0, column=0, sticky=Tkinter.W)
    StatusDisplayScrollbar.config(command=StatusDisplay.yview)
    ButtonState = Tkinter.DISABLED
    OKButton = Tkinter.Button(master=DialogWindow, text='Close Window',
state=ButtonState, default=Tkinter.ACTIVE, command=lambda : CloseWindow())
    OKButton.grid(row=10, column=0, columnspan=2)
    # redirect stdin & stdout
    import sys
    OldStdout = sys.stdout
    StdoutDisplay = StdoutRedirect()
    sys.stdout = StdoutDisplay
    OldStdin = sys.stdin
    StdinSource = StdinRedirect()
    sys.stdin = StdinSource

    # run ESPSearch, trapping the exit from it
    try:
        import espsearch
    except SystemExit:
        ButtonState = Tkinter.NORMAL
        OKButton.config(state=ButtonState)
    # restore stdin & stdout
    sys.stdout = OldStdout
    sys.stdin = OldStdin
#------------------------------------------------------------------------------
def SaveSettings(Fields, SettingsFile=''):
    """Writes espsearch.ini.
    No error checking is performed to validate settings.
    """
    if SettingsFile == '':
        # prompt user for file name
        Diag = tkFileDialog
        OutputFile = Diag.asksaveasfile(filetypes = [('Settings Files', '*.ini'), ('All
Files', '*')], defaultextension='.ini', title='Save Settings')
        # don't try to save if they didn't select a file
        if OutputFile == None: return
    else:
        OutputFile = open(SettingsFile, 'w')
```

```
    # write out all fields, including comments for standard espsearch ini file
    OutputFile.write('# See espsearch.html for information on these settings\n')
    OutputFile.write('# File Locations\n')
    OutputFile.write('Database=' + Fields['Database'].get() + '\n')
    OutputFile.write('Output Directory=' + Fields['Output Directory'].get() + '\n')
    OutputFile.write('Rules File=' + Fields['Rules File'].get() + '\n')
    Fields['Source Directory'].config(state=Tkinter.NORMAL) # active for editing
    OutputFile.write('Source Directory=' + Fields['Source Directory'].get() + '\n')
    Fields['Source Directory'].config(state=Tkinter.DISABLED) # turn this off again
    OutputFile.write('Source File=' + Fields['Source File'].get() + '\n')
    OutputFile.write('\n')
    OutputFile.write('# General Options\n')
    OutputFile.write('Create Directories=' + str(Fields['Create Directories Var'].get())
+ '\n')
    OutputFile.write('Output Format=' + str(Fields['Output Format Var'].get()) + '\n')
    OutputFile.write('Standardize Name Length=' + str(Fields['Name Length Var'].get()) +
'\n')
    OutputFile.write('\n')
    OutputFile.write('# Target Sequence Options\n')
    OutputFile.write('Maximum Mismatches per Segment=' + Fields['Maximum Mismatches per
Segment'].get() + '\n')
    OutputFile.write('Maximum Total Mismatches=' + Fields['Maximum Total
Mismatches'].get() + '\n')
    OutputFile.write('\n')
    OutputFile.write('# Pattern Options\n')
    OutputFile.write('Exact Sequence Pattern Match=' + str(Fields['Pattern Var'].get()) +
'\n')
    OutputFile.write('Pattern=' + Fields['Pattern'].get() + '\n')
    OutputFile.write('\n')
    OutputFile.write('# Performance Options\n')
    OutputFile.write('Buffer=' + Fields['Buffer'].get() + '\n')
    OutputFile.write('RAM Warning Level=' + Fields['RAM Warning Level'].get() + '\n')
    OutputFile.close()
#--------------------------------------------------------------------------
# MAIN
import Tkinter # Load Tkinter GUI module
import tkFileDialog # Tkinter FileDialog
import tkSimpleDialog # for prompt dialogs
SettingsFile = 'espsearch.ini' # the basic user settings file
root = Tkinter.Tk() # define master window
root.title('ESPSearchGUI')
Fields = DrawEverything(root, SettingsFile)
# load espearch.ini & fill fields
root.wm_resizable(0,0) # prevent resizing of window
root.mainloop() # trigger display of window & event handling
```

APPENDIX C
ESPSEARCH SAMPLE FILES

C.1 espsearch.ini

```
# See espsearch.html for information on these settings
# File Locations
Database=humanzif.txt
Output Directory=
Rules File=dna.txt
Source Directory=
Source File=source.txt

# General Options
Create Directories=0
Output Format=1
Standardize Name Lengths=0

# Target Sequence Options
Maximum Mismatches per Segment=0
Maximum Total Mismatches=0

# Pattern Options
Exact Sequence Pattern Match=0
Pattern=ABC

# Performance Options
Buffer=1000
RAM Warning Level=50
```

C.2 humanzif.txt

```
CSNR1=GAA,GAC,GAG
CSNR2=GAA,GAC,GAG
DSAR=ATC,GTC
DSCR=GCC
HSNK=AAC,GAC
HSSR=GTT
ISNR=GAA,GAT
ISNV=AAT
KSNR=GAG
QAHR=GGA
QFNR=GAG
QGNR=GAA
QSHR1=AGA,GGA
QSHR2=GGA
QSHR3=AGA,GGA
QSHR4=GGA
QSHR5=AGA,CGA,GGA,GAA
QSHT=AGA,CGA,TGA
```

```
QSHV=AGA,CGA,TGA
QSNI=AAA,CAA
QSNR1=GAA
QSNR2=GAA
QSNR3=GAA
QSNR4=GAA
QSNK=AAA,GAA,TAA
QSNT=AAA
QSNV1=AAA,CAA
QSNV2=AAA,CAA,GAA,TAA
QSNV3=AAA,CAA
QSNV4=AAA,CAA
QSSR1=GCA,GTA
QSSR2=GCA,GTA
QSSR3=GCA,GTA
QSTR=GCA,GTA
QSTV=GAA,GAC,GAG
QTHQ=AGA,CGA,TGA
QTHR1=AGA,CGA,GGA,GAA
QTHR2=GGA
RDER1=GAG,GCG,GTG
RDER2=GCG
RDER3=GCG
RDER4=GCG
RDER5=GCG
RDER6=GCG
RDHR1=GGG
RDHR2=GGG
RDHT=AGG,CGG,GGG,TGG
RDKI=GGG
RDKR=AGG,GGG
RSHR=GGG
RSNR=GAG
SSNR=GAG
VSNV=AAT
VSSR=GCA,GCG,GCT,GTA,GTG,GTT
VSTR=GCA,GCT
WSNR=GGA,GGT
```

C.3 baxsirna.txt

```
A = gcggcggtgatggacgggtcc
B = atgggggggaggcacccgag
C = aggatgattgccgccgtggac
D = agcaaactggtgctcaaggcc
E = tgggtgagactcctcaagcct
F = tggtgccctctccccatcttc
```

C.4 lxxll.txt

```
class1 = SRLXXLL
class2 = HPLLXXLL
class3 = UJLXXLL
```

```
general1 = LXXLL
```

C.5 dna.txt

```
# matches standard IUPAC DNA base codes & wildcards, will not match wildcards in source
Complements:
A=T, C=G, G=C, T=A, M=K, K=M, R=Y, W=W, S=S, Y=R, B=V, V=B, D=H, H=D, N=N

Mismatches:
A=B, C=D, G=H, T=V, M=K, K=M, R=Y, W=S, S=W, Y=R, B=A, D=C, H=G, V=T, N=

Wildcards:
A=A, C=C, G=G, T=T, M=AC, R=AG, W=AT, K=GT, S=CG, Y=CT, B=CGT, D=AGT, H=ACT, V=ACG,
N=ACGT
```

C.6 dna2.txt

```
# matches standard IUPAC DNA base codes & wildcards, will match wildcards in source
Complements:
A=T, C=G, G=C, T=A, M=K, K=M, R=Y, W=W, S=S, Y=R, B=V, V=B, D=H, H=D, N=N

Mismatches:
A=B, C=D, G=H, T=V, M=K, K=M, R=Y, W=S, S=W, Y=R, B=A, D=C, H=G, V=T, N=

Wildcards:
A=A, C=C, G=G, T=T, M=ACM, R=AGR, W=ATW, K=GTK, S=CGS, Y=CTY, B=CGTB, D=AGTD, H=ACTH,
V=ACGV, N=ACGTNMKRYWSBDHV
```

C.7 rna.txt

```
# matches standard IUPAC RNA base codes & wildcards, will not match wildcards in source
Complements:
A=U, C=G, G=C, U=A, M=K, K=M, R=Y, W=W, S=S, Y=R, B=V, V=B, D=H, H=D, N=N

Mismatches:
A=B, C=D, G=H, U=V, M=K, K=M, R=Y, W=S, S=W, Y=R, B=A, D=C, H=G, V=U, N=

Wildcards:
A=A, C=C, G=G, U=U, M=AC, R=AG, W=AU, K=GU, S=CG, Y=CU, B=CGU, D=AGU, H=ACU, V=ACG,
N=ACGU
```

C.8 rna2.txt

```
# matches sUandard IUPAC RNA base codes & wildcards, will match wildcards in source
Complements:
A=U, C=G, G=C, U=A, M=K, K=M, R=Y, W=W, S=S, Y=R, B=V, V=B, D=H, H=D, N=N

Mismatches:
A=B, C=D, G=H, U=V, M=K, K=M, R=Y, W=S, S=W, Y=R, B=A, D=C, H=G, V=U, N=

Wildcards:
A=A, C=C, G=G, U=U, M=ACM, R=AGR, W=AUW, K=GUK, S=CGS, Y=CUY, B=CGUB, D=AGUD, H=ACUH,
V=ACGV, N=ACGUNMKRYWSBDHV
```

C.9 protein.txt

```
# matches standard IUPAC amino acid codes & wildcards, will not match wildcards in source
Complements:

Mismatches:
A=a, B=b, C=c, D=d, E=e, F=f, G=g, H=h, I=i, K=k, L=l, M=m, N=n, P=p, Q=q, R=r, S=s, T=t,
V=v, W=w, X=, Y=y, Z=z

Wildcards:
A=A, B=DN, C=C, D=D, E=E, F=F, G=G, H=H, I=I, K=K, L=L, M=M, N=N, P=P, Q=Q, R=R, S=S,
T=T, V=V, W=W, Y=Y, Z=EQ,
X=ACDEFGHIKLMNPQRSTVWY, a=CDEFGHIKLMNPQRSTVWY, c=ADEFGHIKLMNPQRSTVWY,
d=ACEFGHIKLMNPQRSTVWY, e=ACDFGHIKLMNPQRSTVWY,
f=ACDEGHIKLMNPQRSTVWY, g=ACDEFHIKLMNPQRSTVWY, h=ACDEFGIKLMNPQRSTVWY,
i=ACDEFGHKLMNPQRSTVWY, k=ACDEFGHILMNPQRSTVWY,
l=ACDEFGHIKMNPQRSTVWY, m=ACDEFGHIKLNPQRSTVWY, n=ACDEFGHIKLMPQRSTVWY,
p=ACDEFGHIKLMNQRSTVWY, q=ACDEFGHIKLMNPRSTVWY,
r=ACDEFGHIKLMNPQSTVWY, s=ACDEFGHIKLMNPQRTVWY, t=ACDEFGHIKLMNPQRSVWY,
v=ACDEFGHIKLMNPQRSTWY, w=ACDEFGHIKLMNPQRSTVY,
y=ACDEFGHIKLMNPQRSTVW
```

C.10 protein2.txt

```
# matches standard IUPAC amino acid codes & wildcards, will match wildcards in source
Complements:

Mismatches:
A=a, B=b, C=c, D=d, E=e, F=f, G=g, H=h, I=i, K=k, L=l, M=m, N=n, P=p, Q=q, R=r, S=s, T=t,
V=v, W=w, X=, Y=y, Z=z

Wildcards:
A=A, B=DNB, C=C, D=D, E=E, F=F, G=G, H=H, I=I, K=K, L=L, M=M, N=N, P=P, Q=Q, R=R, S=S,
T=T, V=V, W=W, Y=Y, Z=EQZ,
X=ACDEFGHIKLMNPQRSTVWYXBZ, a=CDEFGHIKLMNPQRSTVWY, c=ADEFGHIKLMNPQRSTVWY,
d=ACEFGHIKLMNPQRSTVWY, e=ACDFGHIKLMNPQRSTVWY,
f=ACDEGHIKLMNPQRSTVWY, g=ACDEFHIKLMNPQRSTVWY, h=ACDEFGIKLMNPQRSTVWY,
i=ACDEFGHKLMNPQRSTVWY, k=ACDEFGHILMNPQRSTVWY,
l=ACDEFGHIKMNPQRSTVWY, m=ACDEFGHIKLNPQRSTVWY, n=ACDEFGHIKLMPQRSTVWY,
p=ACDEFGHIKLMNQRSTVWY, q=ACDEFGHIKLMNPRSTVWY,
r=ACDEFGHIKLMNPQSTVWY, s=ACDEFGHIKLMNPQRTVWY, t=ACDEFGHIKLMNPQRSVWY,
v=ACDEFGHIKLMNPQRSTWY, w=ACDEFGHIKLMNPQRSTVY,
y=ACDEFGHIKLMNPQRSTVW
```

C.11 protein_ex.txt

```
# matches standard IUPAC amino acid codes & wildcards, will not match wildcards in source
# adds custom wildcards J & U
Complements:

Mismatches:
A=a, B=b, C=c, D=d, E=e, F=f, G=g, H=h, I=i, K=k, L=l, M=m, N=n, P=p, Q=q, R=r, S=s, T=t,
V=v, W=w, X=, Y=y, Z=z, J=j, U=u

Wildcards:
```

A=A, B=DN, C=C, D=D, E=E, F=F, G=G, H=H, I=I, K=K, L=L, M=M, N=N, P=P, Q=Q, R=R, S=S, T=T, V=V, W=W, Y=Y, Z=EQ,

X=ACDEFGHIKLMNPQRSTVWY, a=CDEFGHIKLMNPQRSTVWY, c=ADEFGHIKLMNPQRSTVWY, d=ACEFGHIKLMNPQRSTVWY, e=ACDFGHIKLMNPQRSTVWY,

f=ACDEGHIKLMNPQRSTVWY, g=ACDEFHIKLMNPQRSTVWY, h=ACDEFGIKLMNPQRSTVWY, i=ACDEFGHKLMNPQRSTVWY, k=ACDEFGHILMNPQRSTVWY,

l=ACDEFGHIKMNPQRSTVWY, m=ACDEFGHIKLNPQRSTVWY, n=ACDEFGHIKLMPQRSTVWY, p=ACDEFGHIKLMNQRSTVWY, q=ACDEFGHIKLMNPRSTVWY,

r=ACDEFGHIKLMNPQSTVWY, s=ACDEFGHIKLMNPQRTVWY, t=ACDEFGHIKLMNPQRSVWY, v=ACDEFGHIKLMNPQRSTWY, w=ACDEFGHIKLMNPQRSTVY,

y=ACDEFGHIKLMNPQRSTVW, J=ILV, j=ACDEFGHKMNPQRSTWY, U=ST, u=ACDEFGHIKLMNPQRVWY

APPENDIX D
ESPSEARCH MANUAL

D.1 Introduction

D.1.1 What is ESPSearch?

ESPSearch, which stands for Exact Sequence and Pattern Search, searches a DNA, RNA, or protein sequence for specific target sequences, such as the 3 base triplets recognized by specific zinc fingers. An unlimited number of target sequences can be searched for simultaneously. Pattern matching can also be performed, to look for patterns of hits, such as identifying direct repeats or other long stretches that are recognized by zinc fingers.

ESPSearch is written in Python (http://www.python.org). Python is a scripting language, so the program is source code rather than a compiled program; this means you are free to edit the code if you desire (see Section D.4.3). Because Python can run on essentially any platform, ESPSearch should work on essentially any computer.

If you publish results obtained using ESPSearch, please cite:
Watt, T. J. and Doyle, D. F. (2005) ESPSearch: A Program for Finding Exact Sequences and Patterns in DNA, RNA, or Protein. *Biotechniques, 38*, 109-115.

ESPSearch and related files may be downloaded from http://web.chemistry.gatech.edu/~doyle/espsearch/.

D.1.2 What ESPSearch is Designed to Do

ESPSearch is designed to identify essentially any target sequence within any source sequence to find the exact desired sequence(s). Moreover, it does so using a simple interface that is fast and easy to configure, works

with nearly any operating system, and can be modified if the need arises for additional functionality. Specific abilities include:

- Search arbitrary source sequences for many, possibly complex, target sequences simultaneously. ESPSearch can search for 1 or 1000 target sequences simultaneously with little or no difference in speed, and is unique in its ability to do so with arbitrarily complex sequences.
- Identify complex target sequences that may be difficult or impossible to identify with other tools (e.g., BLAST) due to length restrictions, wildcard constraints, variable regions, or specific mismatch levels. ESPSearch has no length restrictions, and a great deal of complexity may be specified for target sequences.
- Analyze target hits for specific patterns. ESPSearch is unique in its ability to analyze the relationship of target hits according to user-specified patterns constructed from target sequence hits. Patterns can be simple or complex as needed.
- Provide detailed output, including hit locations, mismatches, and size of variable regions. ESPSearch can provide an annotated sequence indicating where, and in which frame, all target sequences are found, or generate a separate file for each target sequence listing hits.
- Allow customization of search parameters at all levels. For example, it is possible to search for non-standard DNA bases, arbitrary groups (e.g., hydrophobic amino acids), or use non-standard complements (see Section D.2.4).

D.1.3 How ESPSearch is Different

ESPSearch is a general tool for identifying any target within any source sequence with complete control over all aspects of the search, but

may not be the best choice for specific applications where specialized tools exist for the search.

- ESPSearch is not a general alignment tool such as BLAST. BLAST is generally faster and more efficient for aligning most sequences than ESPSearch. However, ESPSearch will locate sequences that BLAST cannot find, such as very short sequences and sequences containing many wildcards or gaps.
- ESPSearch can, like many other tools in existence (such as the TRANSFAC tools), identify a specific class of binding site (e.g., transcription factors). However, ESPSearch is not limited to a particular data set or scanning a particular DNA sequence. On the other hand, these other specialized tools may be faster for certain specific applications that take advantage of their specialization.
- ESPSearch differs from most software that search "patterns" (such as TRANSFAC's "Patch" program) in that the "patterns" searched by ESPSearch are combinations of specified target sequences, determined entirely by you. Most other programs are highly restrictive in their "patterns." Moreover, complex pattern searches in ESPSearch are very straightforward through the use of specific sequences in a database and one or more patterns that arrange them as necessary.

D.1.4 Running ESPSearch

First download and install Python (http://www.python.org), version 2.3 or greater (ESPSearch makes use of commands not available in prior versions). Follow the instructions provided on the website for installation; Windows users should find a convenient installation file which will also set up Python scripts (including ESPSearch) to be run by double-clicking the file icon. Users of other operating systems should add Python to their operating

system path, or else will need to run ESPSearch as "python espsearch.py" from the Python directory to invoke ESPSearch. Note that the Python interpreter may cause firewall software to ask if Python may have internet access; this is a normal Python process that you should allow (Python never actually accesses the network).

Then configure **espsearch.ini** as you wish (see Section D.2.1), create a target sequence database (see Section D.2.2), create or download a nucleic acid or protein sequence to search, and create any necessary directories. Then run **espsearch.py**. Note that if ESPSearch ever exits without prompting you to press ENTER or a general Python error is raised (as indicated by a detailed output that contains no clear indication of how to fix the problem), you have provided ESPSearch with a configuration or input that it is unable to handle. ESPSearch can handle common or simple problems gracefully, but is not designed to handle every possible issue.

D.2 Search Configuration

D.2.1 ESPSearch.ini

The ESPSearch configuration file, **espsearch.ini**, contains all search parameters and program options. **espsearch.ini** must be in the same directory as **espsearch.py**.

Two important notes:

- If multiple lines of the same type (e.g., Source File =) are present, only the data from the last (lowest in file) will be remembered.
- Placing "#" before a line will cause ESPSearch to ignore it, allowing you to temporarily "disable" a line.

D.2.1.1 File Locations

These settings indicate where input files reside and output should be directed.

- **Database File** is the file name, and path if not in the current directory, for the file that contains the target sequences to be searched for. See Section D.2.2.
- **Output Directory** is the directory that output files will be created in. Note that this directory must already exist. A blank entry will be interpreted as the current directory. If ESPSearch detects files that will be overwritten in the directory, you will be prompted for confirmation that you wish to overwrite existing files.
- **Rules File** is the file name, and path if not in the current directory, for the file that contains the rules for the search (see Section D.2.4).
- **Source Directory** is the directory containing the Source File(s). If searching multiple Source Files, all files must be in the this directory. A blank entry will be interpreted as the current directory.
- **Source File** is a list of the files to be searched. Multiple files should be separated by commas, and requires that Create Directories (see Section D.2.1.2) is set to 1. See Section D.2.2 for more information.

D.2.1.2 General Options

These settings allow control over global features of the search.

- **Create Directories** defaults to 0. ESPSearch has the ability to create directories within the Output Directory to avoid overwriting files and enable searching multiple Source Files at once. However, the "os" module in Python required for these operations is functional only for Windows, Macintosh, and UNIX operating systems as of this writing. Set Create Directories to 1 if you are using one of these operating systems and wish to have ESPSearch create directories based on the source file names. The effect of setting this to 1 for any other operating system is undetermined.
- **Output Format** defaults to 1. See Section D.3.1.

- **Standardize Name Length** defaults to 0. A value of 1 means that ESPSearch will pad the ends of target sequence names with spaces as necessary to make them all the same length, for easier reading of the output in a fixed-width text editor (e.g., Microsoft Notepad).

D.2.1.3 Target Sequence Options

These settings control mismatch behavior of target sequences.

- **Maximum Mismatches per Segment** defaults to 0, and indicates the maximum number of mismatches you wish to allow in each segment of a target sequence for it to be considered valid. See Section D.3.1.
- **Maximum Total Mismatches** defaults to 0, and indicates the maximum number of mismatches you wish to allow in a hit for it to be considered valid. This value must be at least as large as Maximum Mismatches per Segment. See Section D.3.1.

D.2.1.4 Pattern Options

These settings control the pattern matching behavior.

- **Exact Sequence Pattern Match** defaults to 0, and indicates whether a pattern hit is determined by the literal sequence or by the target sequences that recognize the source sequence. See Section D.2.5.2.
- **Pattern** contains the pattern, or a list of patterns separated by commas, that you wish to match. Leave this line blank if you do not wish to match a pattern. See Section D.2.5.

D.2.1.5 Performance Options

These settings modify the overall performance of ESPSearch, but generally can be left at the default values.

- **Buffer** defaults to 1000. It determines the number of positions read

at a time from the Source File. Lowering this value will usually lead to somewhat slower performance, but may also reduce the amount of RAM used; increasing it may have the opposite effect. Note that if a target sequence or pattern is longer than this value, it will automatically increase to compensate.

- **RAM Warning Level** defaults to 50. This is a qualitative indicator for when ESPSearch will issue a warning about the predicted amount of RAM to be used. This may be adjusted up or down as necessary to increase or decrease the warning threshold. This indicator is the RAM used in addition to the base 6-10 MB required of all searches (mostly by Python itself).

D.2.2 Source Sequences

ESPSearch accepts plain-, FASTA-, EMBL-, or GenBank-formatted source sequences. Plain formatted files should contain only a single strand of the sequence to be searched (if DNA or RNA) and no headers; all other formats must meet standard requirements, except that either lower or upper case characters are accepted (everything is converted to upper case when read in). Blank lines, line numbers, and internal spaces are removed from the sequence when it is read. If multiple sequences are present in the file (allowed for all but plain format), the descriptor line for each sequence containing at least one match will be written into the output file(s). Note that sequences containing no matches will not have their information output, but they have been searched. Note also that a > will be placed at the beginning of the descriptor line, even if the source is EMBL or GenBank format. All other file formats, or incorrectly formatted files of the acceptable varieties, will be assumed to be plain-formatted and may generated unexpected output. All files should contain occasional line breaks to avoid loading the entire file at once, possibly leading to excessive use of RAM.

D.2.3 Target Sequence Databases

Target sequences are loaded from a user-specified database file. This file should contain only the target sequences, but all lines not containing an "=" will be disregarded. Target sequences are entered according to the following format:

- Name of first sequence = First sequence
- Name of second sequence = Second sequence1, Second sequence2, etc.
- etc.

If the same (case-insensitive) name is used more than once, ESPSearch will recognize that two or more sequences have the same name, and assume that they represent a single sequence that has multiple targets. Multiple targets may also be listed directly, by separating sequences with commas following the =. Acceptable characters are determined by the rules file (see Section D.2.4).

Gaps may be introduced into target sequences (but may not begin or end a sequence). Gaps are variable regions of the specified length, and are designated by the format "[minimum length:maximum length]" in the appropriate place in the target sequence, where lengths must be non-negative integers (anything else is physically meaningless). Examples:

- First = AAA[0:2]GGG is equivalent to searching for AAAGGG, AAANGGG, and AAANNGGG simultaneously (and more efficiently)
- Second = AAA[0:0]AAA is equivalent to searching for AAAAAA (usually less efficiently than if no gap is present)
- Third = AAA[0:1]CCC[0:1]GGG is equivalent to searching for AAACCCGGG, AAANCCCGGG, AAACCCNGGG, and AAANCCCNGGG simultaneously
- Fourth = [0:5]AAA[0:5]AAA is invalid due to the initial gap.

The use of gaps creates "segments" within each target sequence, where a "segment" refers to the group of bases marked by a gap. For example, AAA and GGG are the two segments in First. The behavior of ESPSearch with respect to mismatches is affected slightly by the presence of segments. If only one segment is present (i.e., no gaps are used), the settings Maximum Mismatches per Segment and Maximum Total Mismatches are identical: the number of mismatches allowed in a target sequence. If more than one segment exists, these settings allow you to tune the mismatch behavior of the target sequences. Maximum Total Mismatches always refers to the maximum allowable mismatches in the entire target sequence. However, if Maximum Mismatches per Segment is set lower than Maximum Total Mismatches, the distribution of mismatches throughout the target sequence will be spread out. An example of where this is useful is when target sequences represent 2 monomers brought together into a dimer with some spacing (e.g., AAAA[0:2]GGGG). If each monomer can tolerate 1 mismatch, but the dimer can tolerate 2, setting Maximum Mismatches per Segment to 1 and Maximum Total Mismatches to 2 will produce the desired behavior.

A single target may only have 64 explicit sequences assigned to it. This means that "Target = NNNN" is fine, but Target = "AAAA, AAAC, AAAG, ..., TTTC, TTTG, TTTT" will lead to an error.

D.2.4 Search Rules

The rules file contains the characters expected to be found in the target and source sequences, and how they relate to each other. This file is divided into 3 sections, which should consist of a header line ("Complements:", "Mismatches:", or "Wildcards:"), with a list of corresponding information below each header.

- **Complements** contains the expected complement character

relationships for nucleic acid searches. For protein searches, this area should be left blank (although the "Complements:" line must still exist). For example, to set the complement of "A" to "T" and vice versa, enter "A=T, T=A" (without quotes) below the "Complements:" line. All characters that you want to be interpreted as valid should be included in this list. All characters used here must be uppercase (lowercase letters will lead to missed hits when all targets & source sequences are converted to uppercase).

- **Mismatches** contains the single value representing "not" some position. For example, the IUPAC DNA representation for "not A" is "B." Therefore, enter "A=B" here for a standard nucleic acid mismatch. All letters used in this section must exist in the Wildcards section as well. Lowercase letters can be used here. It is very important that all characters present in the Complements section are also present here on the left side of each relationship. For protein searches, this section must contain all the valid characters you expect and their mismatches; if a character you wish to be valid is not listed in this section, ESPSearch will not recognize it as a valid character.
- **Wildcards** contains the "expansion" of all mismatch characters into equivalent characters in the Complements list. For example, "B" would be set to "B=CGT" to represent "not A." If you wish to be able to match wildcards in the source sequence, the Wildcards definitions must include themselves (i.e., to match "B" in the source sequence, use "B=BCGT"). Moreover, to match those wildcards, the same character must be used in the source sequence (i.e., ESPSearch cannot determine that Target=CT,GT,TT is the same as Target=BT, and they are in fact non-equivalent if you are trying to

match wildcards in the source sequence). Note that while lowercase letters are allowed on the left of an "=" (corresponding to a particular Mismatch character), they are not permitted on the right side (they will simply never lead to a match).

All sets of rules may also contain non-letter characters, because case is irrelevant. However, numbers should be avoided, because numbers on the ends of lines will be removed (ESPSearch cannot distinguish a number that represents something from a line number).

The ESPSearch website has a selection of rules files for standard DNA, RNA, and protein searches based on IUPAC definitions.

D.2.5 Patterns

Patterns are a technique for analyzing arrangements of target sequence hits. For example, if you are searching for 10 target sequences, but really want to know when some combination of 3 of those target sequences occur together, entering the appropriate pattern in **espsearch.ini** allows you to gather the desired information.

D.2.5.1 Entering Patterns

Patterns are entered as a series of letters, with gaps placed where appropriate. The following general rules apply:

- Letters are entirely arbitrary. The pattern AA means the same thing as the patterns ZZ and cc. Letters are case sensitive: the pattern Aa is not the same as AA, but is the same as AB.
- Letters do not indicate a particular target sequence, but identical letters mean that the hits in those positions must be the same. For example, the pattern AA means find two identical hits in a row, while ABC finds any three hits in a row.
- Letters that are different mean that hits in those can be, but are not

necessarily, different. For example, the pattern AB will find all hits from the pattern AA, as well as additional hits.

- Gaps are entered in the same manner as within target sequences: [minimum distance:maximum distance]. However, negative values are allowed (indicating overlapping target sequences). Negative values are often not physically meaningful, and will never match unless some targets have overlapping fragments. The magnitude of negative values must be less than the shortest possible target sequence entered (e.g., A[-5:0]B will raise an error if the target AA[0:2]AA is in the database, because the shortest version of that target is only 4 positions). As with target sequences, patterns may not begin or end with a gap.
- Patterns are limited to the letters of the English alphabet (52 characters because of upper and lower case). See Section D.4.3 if you need to change this.

In addition to letters and spaces, additional characters may be used to provide a better defined pattern:

- Placing ! (exclamation point) before a letter means that position in the pattern should be unique. Therefore, any hits in that pattern position that also match hits anywhere else in the pattern will be removed. The pattern A!B matches all the hits from the pattern AB except those also matching AA. Note that using ! ignores any similarity in letters. For example, the pattern A!A and A!B are identical, because the letter following ! is treated as unique, even if it is the same as letters elsewhere in the pattern. Note that if you want all positions to be unique, you must put ! before all (or all but one) of the letters in the pattern.
- > (greater than) and < (less than) mean that all following letters

are to be on the same strand. The patterns AB, >AB, >A>B, A>B, <AB, and <A<B are all equivalent; >A<B is not. ESPSearch assumes an initial > unless another character is provided at the beginning of the sequence.

- | (pipe) is used to indicate that the following hits can be on either strand. The pattern |AB is equivalent to the patterns >AB and >A<B combined.

Note that for protein sequences, <, >, and | are ignored (they are irrelevant).

D.2.5.2 Determining Pattern Behavior

Pattern matching can function in two modes, illustrated by the following examples, which use 2 DNA target sequences, First = AAA and Second = GGG, GGC, and 1 pattern = AA.

- If Exact Sequence Pattern Match is set to 1, pattern matching analyzes the actual source sequences to determine pattern hits. This means that for the pattern to be true for First, the DNA sequence AAAAAA must be encountered. For the pattern to be true for Second, either the sequence GGGGGG or the sequence GGCGGC will produce a hit.
- If Exact Sequence Pattern Match is set to 0, pattern matching looks only at the target sequences to determine pattern hits. So for the pattern AA to be true, the sequence of hits First-First or Second-Second must be encountered. First-First corresponds to the DNA sequence AAAAAA. Second-Second corresponds to 4 DNA sequences: GGGGGG, GGCGGC, GGGGGC, and GGCGGG.

Note that the second case includes all hits from the first case, as well as additional hits. There are two sets of conditions under which the determination of a pattern hit by either method will yield exactly the same

set of results if either (or both) are true:

- The pattern does not contain any letter more than once (i.e., no positions need to be the same target sequence).
- The target sequences each refer to exactly 1 sequence (i.e., no wildcards, gaps, or multiple entries for any target) and there are no mismatches allowed.

If the above conditions are not true, then consider carefully which approach to use. If you are unsure, set Exact Sequence Pattern Match to 0 (the default), because at worst it will find more hits than you expected. To illustrate the difference when a single mismatch is allowed in a target sequence, consider an example using the target sequence AAA and the pattern AA: if Exact Sequence Pattern Match = 1, and 1 mismatch is allowed, the pattern will recognize AAAAAA and AATAAT as hits, but not AATAAA. If Exact Sequence Pattern Match = 0, AATAAA is found by the pattern.

If your target sequences contain gaps, Exact Sequence Pattern Match should always be set to 0, because gaps are by definition variable.

D.2.5.3 Example Patterns

All the following patterns can be entered in multiple equivalent ways; shown is the simplest possible way.

- A = a list of all hits.
- ABCDEF = any sequence of six sequential hits on a single strand.
- AB[0]AB = direct repeat of 2 hits with no gap on the same strand (equivalent to ABAB).
- AB[0:2]<BA = inverted and everted repeats of 2 hits with gaps of 0, 1, or 2 bases.
- A|!B = all sequences of two different sequential hits, regardless of which strand each hit is on, equivalent to A!B and A<!B.

- ABC[3:3]DEF, ABC[3:3]<DEF = any sequence of 3 hits on one strand, a 3 base gap, and any sequence of 3 hits on the same or other strand with a 3 base gap. Note this is different from ABC[3:3]|DEF, because this last pattern also includes patterns such as ABC[3:3]<D>E<F.
- A[0:3]A, A[0:3]A[0:3]A, A[0:3]A[0:3]A[0:3]A = 2, 3, and 4 repeats of hit on the same strand, each separated by 0 to 3 bases.

In many cases, alternate pattern forms will lead to equivalent hits, but presented differently. For example, >AA and <AA will give rise to the same hits, the output files for each pattern will differ in that the strands designated for each hit will be reversed (and therefore the sequences will all be reverse complements).

D.3 Running a Search

D.3.1 Interpreting the Target Sequence Output

All styles of output are tab-formatted for easy loading into spreadsheets for convenience of reading and analysis. Format numbers correspond to the number of files generated: format 0 is undetermined (based on the database used), format 1 produces a single file, and format 2 produces 2 files (for nucleic acids; only 1 for proteins). Output file names are generated from a combination of the database used in the search and the output format.

In all cases, the search output indicates matches from the N-C and 5'-3' (for the strand it is on). To match in the opposite direction, simply reverse your target sequences.

D.3.1.1 Format 0

In this format, every target sequence has its own file for output. Sequences recognized by more than one target sequence are therefore

output to multiple files. Each file consists of the position of each hit (the 5'-most base or most N-terminal amino acid of the sequence), the specific sequence for each hit, the strand that each hit occurs on (+ = sense, - = antisense) if nucleic acid, and the number of mismatches in each hit. In addition, if the target sequence includes gaps, the value (in number of bases) for each gap will be listed.

Note that the position values of - strand hits may not proceed sequentially if one or more gaps exist within the target sequences; this is nothing to worry about, and is simply a reflection of the fact that ESPSearch is actually keeping the 5'-most sense base fixed (i.e., the 3'-most antisense base).

This format is constrained by inherent limits on the number of open files allowed in a given operating system, which varies considerably (typically around 500 on Windows systems and 1000 on Linux). If you are searching for very large numbers of sequences, you may have to switch to Format 1. This format also constrains the names of target sequences to strings that are acceptable as file names on your operating system (e.g., "*" is usually not a valid character in a file name, so don't use it in a target sequence if using this format). Unacceptable characters will cause ESPSearch to crash (error-checking is not performed because it is platform-dependent).

D.3.1.2 Format 1

This format is the default. It produces a single file containing all hits in a similar style as format 0, except that an extra column is inserted at the beginning of the file indicating which target each entry corresponds to.

D.3.1.3 Format 2

In this format, 2 files are generated, corresponding to the sense

strand (defined as the sequence in the source file) and all hits on it, and a second corresponding to the antisense strand and all hits. The sense strand begins (with position 1) at the 5' end and ends with the 3' base. The antisense strand is written to complement the sense strand, meaning that position 1 is the 3' end, and the final base is the 5' end. Therefore, positions for either strand refer to the same base pair.

Within each file, hits are listed according to the frame they occur in, where Frame 1 is defined as the reading frame beginning with the first base. Hits are listed next to the 5'-most base they recognize. This means that in the sense output, hits recognize the base they are listed next to and the following bases going *down*, while in the antisense file hits recognize the base they are listed next to and the following bases going *up*.

For proteins, only a single file is output, annotated from the N- to C-termini in an equivalent manner to the nucleic acid output without reading frames.

This format outputs no information regarding the mismatches in a hit if any exist, nor will it work if any target sequence contains a gap. If using nucleic acid rules, target sequences in the database must also all be the same length. If multiple hits recognize a single position, they will be listed sequentially, separated by a space.

D.3.2 Interpreting the Pattern Output

Each pattern is output to its own file, named using the letters in the pattern, the database, and a pattern number (represented as "p#"). Gaps are represented as "_" in the file name. Each file lists the position (5' most base or most N-terminal amino acid) of the pattern hit, the sequence the hit corresponds to, and the strand the hit occurs on if nucleic acid. Following this information is a breakdown of which target sequences occur in each position within the pattern for the hit, with the appropriate control character

in front. This file is tab-delimited for easy loading into a spreadsheet program. If multiple targets match a particular position in the pattern, they will be listed sequentially, separated by a space. Note that the position values of - strand hits may not proceed sequentially if one or more gaps exist within the target sequences; this is nothing to worry about, and is simply a reflection of the fact that ESPSearch is actually keeping the 5'-most sense base fixed (i.e., the 3'-most antisense base).

D.3.3 Tips for Optimizing Searches

ESPSearch typically scanned 500,000 to 2,500,000 positions/second on a Pentium IV 1.8 GHz PC running Windows 2000. Speeds on a Pentium III 500 MHz PC were about half that. However, it is certainly possible to design a search that runs at only 50,000 positions/second or slower, and a few searches will run faster. The tips below will help keep your searches fast and memory-efficient.

- Run Psyco if available for your computer (see Section D.4.2).
- Use only the mismatches and wildcards needed. For example, try to represent "N" or "X" (for standard DNA/protein searches) as gaps. High mismatch levels and/or extensive use of wildcards (especially when also trying to match wildcards in the source sequence) can rapidly lead to high RAM use and may slow searches somewhat.
- The search speed is linearly dependent on the length of the source sequence and the amount of output generated, and approximately exponentially related to the number of partial target sequences found. To improve speed performance, use the search parameters to avoid unwanted output and consider redesigning target sequences to minimize wildcards and mismatches. Moreover, the length of the shortest first segment (and last if a nucleic acid search) is especially important; try to keep these sequences long enough and sufficiently

non-variable to avoid finding many partial hits. For example, if searching for One=AAA and Two=GNNGGGGGGG, 17 out of every 64 positions will lead to a partial hit: AAA & GNN.

- Avoid exceeding physical free RAM. Use of virtual memory is much slower and may lead to an effective halt of the search.
- Patterns are usually run quickly, but must run sequentially. Therefore, only search for the desired patterns, and try to combine patterns if possible (e.g., A>B and A<B can be combined to A|B to improve speed). In addition, try to design target sequences so gaps with a large difference in minimum and maximum value (approximately 50 or so) are placed into patterns rather than target sequences; patterns generally handle gaps much faster.
- Initialization time is usually less than 1 second. However, initialization time is roughly linearly proportional to RAM use, so performing a search that uses several hundred MB of RAM may take up to a minute to initialize. Initialization also takes longer for Output Format 0, but this is only noticeable when using a database containing a few hundred or more target sequences.

D.4 Inner Workings

D.4.1 Algorithms in ESPSearch

After ESPSearch loads the target sequence database, it generates all variations of target sequences due to wildcards and mismatches, and each sequence is assigned a unique memory address. Reverse complements to each sequence are also generated for nucleic acid searches, to identify matches on the complement strand without actually generating or searching the complement. ESPSearch reads the source sequence (the sequence to be searched) with a "window" that corresponds to the target sequence length; if

the current "window" sequence matches any sequence stored in memory, the appropriate output is generated. For patterns, ESPSearch assigns the first position of the pattern to the current hit, then searches forward through the pattern to determine if hits exist in the positions designated by the pattern. For example, to match the pattern ABAB, ESPSearch determines if a hit was found at the position immediately after the first hit, then if a hit identical to the first was found immediately after the second, and if a hit identical to the second was found immediately after the third, all on the same strand. Target sequences containing gaps are found in a method similar to patterns (i.e., ESPSearch identifies the target sequence one section at a time, only generating output when the entire sequence is located).

D.4.2 Psyco Acceleration

The optional "psyco" module for Python (available at http://psyco.sourceforge.net) can greatly accelerate the speed of ESPSearch. However, it is only available for Intel-386 compatible processors (386, 486, Pentium, AMD K6 & K7, etc.) as of this writing. Psyco will generally increase memory usage somewhat (2-5 megabytes), but will provide a boost of approximately 3-fold in processing time. Simply install Psyco according to its installation instructions and ESPSearch will automatically use it.

D.4.3 Modifying ESPSearch

ESPSearch has been extensively optimized for speed, but so as to allow for a wide range of inputs. It is certainly possible to refine the script to suit some specific need and improve performance. In general, the limiting factor is the name lookup (dictionary hash call in particular), which is a very fast C routine, so to generally improve performance, avoid dictionary calls. If you do modify this program, you must do so according to the terms of the GNU General Public License (http://www.gnu.org/licenses/old-licenses/gpl-

2.0.html).

Specific sections of the code may be of interest for certain minor changes, and these are outlined below.

- Initialization file name: this is set by the "SettingsFile" variable in the main area, about line 1125.
- Output file names: these are assigned in 2 locations in the LoadDatabase routine, about line 900 (several lines) and line 1000.
- Adding source file formats: to add another format as acceptable input, lines checking for the appropriate indicators in the file should be added to the lines checking for current formats. These are in the FindSequences routine in two locations: about line 360 (to do an initial check to skip the first headers) and line 380 (for noting the end of a sequence to skip any following headers).
- RAM Warning Level conversion factor: this is a simple conversion to make the number for RAM Warning Level roughly equivalent to MB of RAM. This was calibrated under particular conditions, and if you wish to recalibrate, change constant in the "RAMThreshold = " line in the LoadDatabase routine, about line 860.
- Additional pattern characters: adding characters to the "PatternChars" string in DefineFunction, about line 70, will allow you to use additional characters to indicate pattern positions. Avoid using >, < |, !, [, and], because they will be treated as control characters even if they are in PatternChars.

APPENDIX E
ESPSEARCHGUI MANUAL

E.1 Introduction

E.1.1 What is ESPSearchGUI?

ESPSearchGUI, which stands for Exact Sequence and Pattern Search Graphical User Interface, is a user interface front-end for ESPSearch. If you don't know what ESPSearch is, you probably don't need to be using this program.

ESPSearchGUI is designed simplify your interactions with ESPSearch by creating an interactive environment for configuring searches. ESPSearchGUI also allows you to save settings for later use, making management of different settings sets easier. Finally, ESPSearchGUI is a graphical user interface, which in and of itself seems to make many people happy (and there's nothing wrong with that).

ESPSearchGUI does NOT add ANY functionality to ESPSearch. This is merely a settings management tool, and requires **espsearch.py** to be present in the same directory to actually perform any searches.

Version 1.01 is designed to work with version 1.01 of ESPSearch, although it will also work with v1.00. ESPSearchGUI should work with any operating system that supports TCl/Tk (Unix, Linux, and Windows, and some others) and has Python installed. However, at this stage it has not been thoroughly tested. USE THIS PROGRAM AT YOUR OWN RISK (but if it loads, it probably will work).

E.1.2 Installing and Running

Simply make sure that **espsearchgui.py** is in the same directory as **espsearch.py** (the ESPSearch Python file), and run **espsearchgui.py**

instead of **espsearch.py**. Python (version 2.3 or later) must also be installed on the system. When you start ESPSearch, the default settings will be loaded.

E.2 Loading and Saving

E.2.1 Loading Settings Files

The "Load Settings" button will open a dialog window to select a file. ESPSearchGUI performs no checking to confirm that the file is actually an ESPSearch configuration file, so choosing random files may cause ESPSearchGUI to crash. If the settings file is invalid or incomplete, the default settings will be inserted as needed.

E.2.2 Saving Settings Files

The "Save Settings" button allows you to save the current settings to a file other than **espsearch.ini**. These settings will not be used by ESPSearch unless you load the settings and run ESPSearch through ESPSearchGUI unless you save the settings as **espsearch.ini**. You do NOT need to save the settings before running ESPSearch, as **espsearch.ini** will automatically be written, but you do need to save the settings if you wish to keep them for later use. Note that ESPSearchGUI does not check to ensure the settings you have provided are valid.

E.3 Running ESPSearch

E.3.1 Starting a Search

The "Run Search" button will write the current settings (as displayed) to **espsearch.ini** (overwriting any existing settings in the file), and run ESPSearch. Note that ESPSearchGUI will not check for any errors in the settings, and relies on ESPSearch's load routines to do so. ESPSearch will run in a window over the settings window, locking the settings until

ESPSearch has completed. If ESPSearch exits normally for any reason (even an error in the settings), you can close this window and return to the settings window. If ESPSearch crashes, you may get locked out of the settings window and be unable to close it. Please send email to us if this occurs with a detailed explanation, but no promises of support are made.

E.3.2 Interacting with ESPSearch

While ESPSearch is running, it may occasionally require feedback (just as if you were running it without ESPSearchGUI). When this happens, a dialog box with an input box will appear. Simply enter the necessary information ("y", "n", or nothing, as appropriate), and press the ENTER key or click the OK button to continue. The CANCEL button or invalid input may cause ESPSearch to repeat its request.

REFERENCES

1. Mangelsdorf, D. J., Ong, E. S., Dyck, J. A., and Evans, R. M. (1990) Nuclear receptor that identifies a novel retinoic acid response pathway, *Nature 345*, 224-229.

2. Phelps, C., Gburcik, V., Suslova, E., Dudek, P., Forafonov, F., Bot, N., MacLean, M., Fagan, R. J., and Picard, D. (2006) Fungi and animals may share a common ancestor to nuclear receptors, *Proc Natl Acad Sci U S A 103*, 7077-7081.

3. Thornton, J. W., Need, E., and Crews, D. (2003) Resurrecting the ancestral steroid receptor: ancient origin of estrogen signaling, *Science 301*, 1714-1717.

4. Nuclear Receptors Committee (1999) A Unified Nomenclature System for the Nuclear Receptor Subfamily, *Cell 97*, 1-20.

5. Germain, P., Staels, B., Dacquet, C., Spedding, M., and Laudet, V. (2006) Overview of nomenclature of nuclear receptors, *Pharmacol. Rev. 58*, 685-704.

6. Francis, G. A., Fayard, E., Picard, F., and Auwerx, J. (2003) Nuclear receptors and the control of metabolism, *Annu. Rev. Physiol. 65*, 261-311.

7. Khorasanizadeh, S., and Rastinejad, F. (2001) Nuclear-receptor interactions on DNA-response elements, *Trends Biochem. Sci. 26*, 384-390.

8. Thompson, E. B., and Kumar, R. (2003) DNA binding of nuclear hormone receptors influences their structure and function, *Biochem. Biophys. Res. Commun. 306*, 1-4.

9. Subauste, J. S., Katz, R. W., and Koenig, R. J. (1994) DNA binding specificity and function of retinoid X receptor alpha, *J Biol Chem 269*, 30232-30237.

10. Yang, Y. Z., Subauste, J. S., and Koenig, R. J. (1995) Retinoid X receptor alpha binds with the highest affinity to an imperfect direct repeat response element, *Endocrinology 136*, 2896-2903.

11. Castelein, H., Janssen, A., Declercq, P. E., and Baes, M. (1996) Sequence requirements for high affinity retinoid X receptor-alpha homodimer binding, *Mol. Cell. Endocrinol. 119*, 11-20.

12. Claessens, F., Verrijdt, G., Schoenmakers, E., Haelens, A., Peeters, B., Verhoeven, G., and Rombauts, W. (2001) Selective DNA binding by the androgen receptor as a mechanism for hormone-specific gene regulation, *J Steroid Biochem Mol Biol 76*, 23-30.

13. Wood, J. R., Likhite, V. S., Loven, M. A., and Nardulli, A. M. (2001) Allosteric modulation of estrogen receptor conformation by different estrogen response elements, *Mol. Endocrinol. 15*, 1114-1126.

14. Hall, J. M., McDonnell, D. P., and Korach, K. S. (2002) Allosteric regulation of estrogen receptor structure, function, and coactivator recruitment by different estrogen response elements, *Mol. Endocrinol. 16*, 469-486.

15. Geserick, C., Meyer, H. A., Barbulescu, K., and Haendler, B. (2003) Differential modulation of androgen receptor action by deoxyribonucleic acid response elements, *Mol. Endocrinol. 17*, 1738-1750.

16. Krieg, A. J., Krieg, S. A., Ahn, B. S., and Shapiro, D. J. (2004) Interplay between estrogen response element sequence and ligands controls in vivo binding of estrogen receptor to regulated genes, *J Biol Chem 279*, 5025-5034.

17. Tian, H., Mahajan, M. A., Wong, C. T., Habeos, I., and Samuels, H. H. (2006) The N-Terminal A/B domain of the thyroid hormone receptor-beta2 isoform influences ligand-dependent recruitment of coactivators to the ligand-binding domain, *Mol. Endocrinol. 20*, 2036-2051.

18. Morin, B., Nichols, L. A., and Holland, L. J. (2006) Flanking sequence composition differentially affects the binding and functional characteristics of glucocorticoid receptor homo- and heterodimers, *Biochemistry 45*, 7299-7306.

19. Mengeling, B. J., Pan, F., and Privalsky, M. L. (2005) Novel mode of deoxyribonucleic acid recognition by thyroid hormone receptors: thyroid hormone receptor beta-isoforms can bind as trimers to natural response elements comprised of reiterated half-sites, *Mol. Endocrinol. 19*, 35-51.

20. Chen, H., and Privalsky, M. L. (1995) Cooperative formation of high-order oligomers by retinoid X receptors: an unexpected mode of DNA recognition, *Proc Natl Acad Sci U S A 92*, 422-426.

21. Kersten, S., Kelleher, D., Chambon, P., Gronemeyer, H., and Noy, N. (1995) Retinoid X receptor alpha forms tetramers in solution, *Proc Natl Acad Sci U S A 92*, 8645-8649.

22. Heery, D. M., Kalkhoven, E., Hoare, S., and Parker, M. G. (1997) A signature motif in transcriptional co-activators mediates binding to nuclear receptors, *Nature 387*, 733-736.

23. Hatzivassiliou, E., Koukos, G., Ribeiro, A., Zannis, V., and Kardassis, D. (2003) Functional specificity of two hormone response elements present on the human apoA-II promoter that bind retinoid X receptor alpha/thyroid receptor beta heterodimers for retinoids and thyroids:

synergistic interactions between thyroid receptor beta and upstream stimulatory factor 2a, *Biochem J. 376*, 423-431.

24. Takano, Y., Adachi, S., Okuno, M., Muto, Y., Yoshioka, T., Matsushima-Nishiwaki, R., Tsurumi, H., Ito, K., Friedman, S. L., Moriwaki, H., Kojima, S., and Okano, Y. (2004) The RING finger protein, RNF8, interacts with retinoid X receptor alpha and enhances its transcription-stimulating activity, *J Biol Chem 279*, 18926-18934.

25. Lefebvre, P., Gaub, M. P., Tahayato, A., Rochette-Egly, C., and Formstecher, P. (1995) Protein phosphatases 1 and 2A regulate the transcriptional and DNA binding activities of retinoic acid receptors, *J Biol Chem 270*, 10806-10816.

26. Lee, H. Y., Suh, Y. A., Robinson, M. J., Clifford, J. L., Hong, W. K., Woodgett, J. R., Cobb, M. H., Mangelsdorf, D. J., and Kurie, J. M. (2000) Stress pathway activation induces phosphorylation of retinoid X receptor, *J Biol Chem 275*, 32193-32199.

27. Harish, S., Ashok, M. S., Khanam, T., and Rangarajan, P. N. (2000) Serine 27, a human retinoid X receptor alpha residue, phosphorylated by protein kinase A is essential for cyclicAMP-mediated downregulation of RXRalpha function, *Biochem. Biophys. Res. Commun. 279*, 853-857.

28. Matsushima-Nishiwaki, R., Okuno, M., Adachi, S., Sano, T., Akita, K., Moriwaki, H., Friedman, S. L., and Kojima, S. (2001) Phosphorylation of retinoid X receptor alpha at serine 260 impairs its metabolism and function in human hepatocellular carcinoma, *Cancer Res. 61*, 7675-7682.

29. Zhao, Y., Qin, S., Atangan, L. I., Molina, Y., Okawa, Y., Arpawong, H. T., Ghosn, C., Xiao, J., Vuligonda, V., Brown, G., and Chandraratna, R. A. S. (2004) Casein kinase 1alpha interacts with retinoid X receptor and interferes with agonist-induced apoptosis, *J Biol Chem 279*, 30844-30849.

30. Bruck, N., Bastien, J., Bour, G., Tarrade, A., Plassat, J., Bauer, A., Adam-Stitah, S., and Rochette-Egly, C. (2005) Phosphorylation of the retinoid x receptor at the omega loop, modulates the expression of retinoic-acid-target genes with a promoter context specificity, *Cell. Signal. 17*, 1229-1239.

31. Sun, K., Montana, V., Chellappa, K., Brelivet, Y., Moras, D., Maeda, Y., Parpura, V., Paschal, B. M., and Sladek, F. M. (2007) Phosphorylation of a conserved serine in the deoxyribonucleic acid binding domain of nuclear receptors alters intracellular localization, *Mol. Endocrinol. 21*, 1297-1311.

32. Zimmerman, T. L., Thevananther, S., Ghose, R., Burns, A. R., and Karpen, S. J. (2006) Nuclear export of retinoid X receptor alpha in response to interleukin-1beta-mediated cell signaling: roles for JNK and SER260, *J*

Biol Chem 281, 15434-15440.

33. Choi, S. J., Chung, S. S., Rho, E. J., Lee, H. W., Lee, M. H., Choi, H., Seol, J. H., Baek, S. H., Bang, O. S., and Chung, C. H. (2006) Negative modulation of RXRalpha transcriptional activity by small ubiquitin-related modifier (SUMO) modification and its reversal by SUMO-specific protease SUSP1, *J Biol Chem 281*, 30669-30677.

34. Elbi, C., Walker, D. A., Romero, G., Sullivan, W. P., Toft, D. O., Hager, G. L., and DeFranco, D. B. (2004) Molecular chaperones function as steroid receptor nuclear mobility factors, *Proc Natl Acad Sci U S A 101*, 2876-2881.

35. Weinberg, A., Carter, D., Ahonen, M., Alarid, E., Murdoch, F., and Fritsch, M. (2007) The DNA Binding Domain of Estrogen Receptor alpha Is Required for High-Affinity Nuclear Interaction Induced by Estradiol, *Biochemistry 46*, 8933-8942.

36. Prufer, K., and Barsony, J. (2002) Retinoid X receptor dominates the nuclear import and export of the unliganded vitamin D receptor, *Mol. Endocrinol. 16*, 1738-1751.

37. Nieva, C., Gwozdz, T., Dutko-Gwozdz, J., Wiedenmann, J., Spindler-Barth, M., Wieczorek, E., Dobrucki, J., Dus, D., Henrich, V., Ozyhar, A., and Spindler, K. (2005) Ultraspiracle promotes the nuclear localization of ecdysteroid receptor in mammalian cells, *Biol. Chem. 386*, 463-470.

38. Yasmin, R., Williams, R. M., Xu, M., and Noy, N. (2005) Nuclear import of the retinoid X receptor, the vitamin D receptor, and their mutual heterodimer, *J Biol Chem 280*, 40152-40160.

39. Egea, P. F., Klaholz, B. P., and Moras, D. (2000) Ligand-protein interactions in nuclear receptors of hormones, *FEBS Lett. 476*, 62-67.

40. Steinmetz, A. C., Renaud, J. P., and Moras, D. (2001) Binding of ligands and activation of transcription by nuclear receptors, *Annu. Rev. Biophys. Biomol. Struct. 30*, 329-359.

41. Nagy, L., and Schwabe, J. W. (2004) Mechanism of the nuclear receptor molecular switch, *Trends Biochem. Sci. 29*, 317-324.

42. Germain, P., Chambon, P., Eichele, G., Evans, R. M., Lazar, M. A., Leid, M., De Lera, A. R., Lotan, R., Mangelsdorf, D. J., and Gronemeyer, H. (2006) International Union of Pharmacology. LXIII. Retinoid X receptors, *Pharmacol Rev 58*, 760-772.

43. Melvin, V. S., Harrell, C., Adelman, J. S., Kraus, W. L., Churchill, M., and Edwards, D. P. (2004) The Role of the C-terminal Extension (CTE) of the Estrogen Receptor {alpha} and {beta} DNA Binding Domain in DNA Binding and Interaction with HMGB, *J Biol Chem 279*, 14763-14771.

44. Kumar, R., Betney, R., Li, J., Thompson, E. B., and McEwan, I. J. (2004) Induced alpha-helix structure in AF1 of the androgen receptor upon binding transcription factor TFIIF, *Biochemistry 43*, 3008-3013.

45. Kumar, R., Volk, D. E., Li, J., Lee, J. C., Gorenstein, D. G., and Thompson, E. B. (2004) TATA box binding protein induces structure in the recombinant glucocorticoid receptor AF1 domain, *Proc Natl Acad Sci U S A 101*, 16425-16430.

46. Berbaum, J., and Harrison, R. K. (2005) Comparison of full-length versus ligand binding domain constructs in cell-free and cell-based peroxisome proliferator-activated receptor alpha assays, *Anal. Biochem. 339*, 121-128.

47. Rastinejad, F. (2001) Retinoid X receptor and its partners in the nuclear receptor family, *Curr. Opin. Struct. Biol. 11*, 33-38.

48. Miyamoto, T., Kakizawa, T., Ichikawa, K., Nishio, S., Takeda, T., Suzuki, S., Kaneko, A., Kumagai, M., Mori, J., Yamashita, K., Sakuma, T., and Hashizume, K. (2001) The role of hinge domain in heterodimerization and specific DNA recognition by nuclear receptors, *Mol. Cell. Endocrinol. 181*, 229-238.

49. Shaffer, P. L., McDonnell, D. P., and Gewirth, D. T. (2005) Characterization of Transcriptional Activation and DNA-Binding Functions in the Hinge Region of the Vitamin D Receptor, *Biochemistry 44*, 2678-2685.

50. Nascimento, A. S., Dias, S. M., Nunes, F. M., Aparicio, R., Ambrosio, A. L., Bleicher, L., Figueira, A. C., Santos, M. A., de Oliveira Neto, M., Fischer, H., Togashi, M., Craievich, A. F., Garratt, R. C., Baxter, J. D., Webb, P., and Polikarpov, I. (2006) Structural rearrangements in the thyroid hormone receptor hinge domain and their putative role in the receptor function, *J Mol Biol 360*, 586-598.

51. Mangelsdorf, D. J., Borgmeyer, U., Heyman, R. A., Zhou, J. Y., Ong, E. S., Oro, A. E., Kakizuka, A., and Evans, R. M. (1992) Characterization of three RXR genes that mediate the action of 9-cis retinoic acid, *Genes Dev. 6*, 329-344.

52. Oro, A. E., McKeown, M., and Evans, R. M. (1990) Relationship between the product of the Drosophila ultraspiracle locus and the vertebrate retinoid X receptor, *Nature 347*, 298-301.

53. Mark, M., Ghyselinck, N. B., and Chambon, P. (2006) Function of retinoid nuclear receptors: lessons from genetic and pharmacological dissections of the retinoic acid signaling pathway during mouse embryogenesis, *Annu. Rev. Pharmacol. Toxicol. 46*, 451-480.

54. Sucov, H. M., Dyson, E., Gumeringer, C. L., Price, J., Chien, K. R., and Evans, R. M. (1994) RXR alpha mutant mice establish a genetic basis

for vitamin A signaling in heart morphogenesis, *Genes Dev. 8*, 1007-1018.

55. Dyson, E., Sucov, H. M., Kubalak, S. W., Schmid-Schonbein, G. W., DeLano, F. A., Evans, R. M., Ross, J. J., and Chien, K. R. (1995) Atrial-like phenotype is associated with embryonic ventricular failure in retinoid X receptor alpha -/- mice, *Proc Natl Acad Sci U S A 92*, 7386-7390.

56. Kastner, P., Mark, M., Leid, M., Gansmuller, A., Chin, W., Grondona, J. M., Decimo, D., Krezel, W., Dierich, A., and Chambon, P. (1996) Abnormal spermatogenesis in RXR beta mutant mice, *Genes Dev. 10*, 80-92.

57. Krezel, W., Dupe, V., Mark, M., Dierich, A., Kastner, P., and Chambon, P. (1996) RXR gamma null mice are apparently normal and compound RXR alpha +/-/RXR beta -/-/RXR gamma -/- mutant mice are viable, *Proc Natl Acad Sci U S A 93*, 9010-9014.

58. Rowe, A., Eager, N. S., and Brickell, P. M. (1991) A member of the RXR nuclear receptor family is expressed in neural-crest-derived cells of the developing chick peripheral nervous system, *Development 111*, 771-778.

59. Blumberg, B., Mangelsdorf, D. J., Dyck, J. A., Bittner, D. A., Evans, R. M., and De Robertis, E. M. (1992) Multiple retinoid-responsive receptors in a single cell: families of retinoid "X" receptors and retinoic acid receptors in the Xenopus egg, *Proc Natl Acad Sci U S A 89*, 2321-2325.

60. Robertson, K. A., Emami, B., Mueller, L., and Collins, S. J. (1992) Multiple members of the retinoic acid receptor family are capable of mediating the granulocytic differentiation of HL-60 cells, *Mol. Cell. Biol. 12*, 3743-3749.

61. Oro, A. E., McKeown, M., and Evans, R. M. (1992) The Drosophila retinoid X receptor homolog ultraspiracle functions in both female reproduction and eye morphogenesis, *Development 115*, 449-462.

62. Aneskievich, B. J., and Fuchs, E. (1992) Terminal differentiation in keratinocytes involves positive as well as negative regulation by retinoic acid receptors and retinoid X receptors at retinoid response elements, *Mol. Cell. Biol. 12*, 4862-4871.

63. Thaller, C., Hofmann, C., and Eichele, G. (1993) 9-cis-retinoic acid, a potent inducer of digit pattern duplications in the chick wing bud, *Development 118*, 957-965.

64. Downes, M., Mynett-Johnson, L., and Muscat, G. E. (1994) The retinoic acid and retinoid X receptors are differentially expressed during myoblast differentiation, *Endocrinology 134*, 2658-2661.

65. Rusten, L. S., Dybedal, I., Blomhoff, H. K., Blomhoff, R., Smeland, E. B., and Jacobsen, S. E. (1996) The RAR-RXR as well as the RXR-RXR pathway is involved in signaling growth inhibition of human CD34+ erythroid progenitor cells, *Blood 87*, 1728-1736.

66. Minucci, S., Saint-Jeannet, J. P., Toyama, R., Scita, G., DeLuca, L. M., Tiara, M., Levin, A. A., Ozato, K., and Dawid, I. B. (1996) Retinoid X receptor-selective ligands produce malformations in Xenopus embryos, *Proc Natl Acad Sci U S A 93*, 1803-1807.

67. Lee, H. Y., Dawson, M. I., Walsh, G. L., Nesbitt, J. C., Eckert, R. L., Fuchs, E., Hong, W. K., Lotan, R., and Kurie, J. M. (1996) Retinoic acid receptor- and retinoid X receptor-selective retinoids activate signaling pathways that converge on AP-1 and inhibit squamous differentiation in human bronchial epithelial cells, *Cell Growth Differ. 7*, 997-1004.

68. Apfel, C. M., Kamber, M., Klaus, M., Mohr, P., Keidel, S., and LeMotte, P. K. (1995) Enhancement of HL-60 differentiation by a new class of retinoids with selective activity on retinoid X receptor, *J Biol Chem 270*, 30765-30772.

69. Kostrouch, Z., Kostrouchova, M., Love, W., Jannini, E., Piatigorsky, J., and Rall, J. E. (1998) Retinoic acid X receptor in the diploblast, Tripedalia cystophora, *Proc Natl Acad Sci U S A 95*, 13442-13447.

70. Jenkins, S. J., Hutson, D. R., and Kubalak, S. W. (2005) Analysis of the proepicardium-epicardium transition during the malformation of the RXRalpha-/- epicardium, *Dev. Dyn. 233*, 1091-1101.

71. Jones, G., and Sharp, P. A. (1997) Ultraspiracle: an invertebrate nuclear receptor for juvenile hormones, *Proc Natl Acad Sci U S A 94*, 13499-13503.

72. Alvarez, R., Checa, M., Brun, S., Vinas, O., Mampel, T., Iglesias, R., Giralt, M., and Villarroya, F. (2000) Both retinoic-acid-receptor- and retinoid-X-receptor-dependent signalling pathways mediate the induction of the brown-adipose-tissue-uncoupling-protein-1 gene by retinoids, *Biochem J. 345 Pt 1*, 91-97.

73. Hida, T., Tai, K., Tokuhara, N., Ishibashi, A., Kikuchi, K., Hibi, S., Yoshimura, H., Nagai, M., Yamauchi, T., and Kobayashi, S. (2001) Existence of retinoic acid-receptor-independent retinoid X-receptor-dependent pathway in myeloid cell function, *Jpn. J. Pharmacol. 85*, 60-69.

74. Mascrez, B., Mark, M., Krezel, W., Dupe, V., LeMeur, M., Ghyselinck, N. B., and Chambon, P. (2001) Differential contributions of AF-1 and AF-2 activities to the developmental functions of RXR alpha, *Development 128*, 2049-2062.

75. Wiens, M., Batel, R., Korzhev, M., and Muller, W. E. G. (2003) Retinoid X

receptor and retinoic acid response in the marine sponge Suberites domuncula, *J Exp Biol 206*, 3261-3271.

76. Nishikawa, J., Mamiya, S., Kanayama, T., Nishikawa, T., Shiraishi, F., and Horiguchi, T. (2004) Involvement of the retinoid X receptor in the development of imposex caused by organotins in gastropods, *Environ. Sci. Technol. 38*, 6271-6276.

77. Merki, E., Zamora, M., Raya, A., Kawakami, Y., Wang, J., Zhang, X., Burch, J., Kubalak, S. W., Kaliman, P., Belmonte, J. C. I., Chien, K. R., and Ruiz-Lozano, P. (2005) Epicardial retinoid X receptor alpha is required for myocardial growth and coronary artery formation, *Proc Natl Acad Sci U S A 102*, 18455-18460.

78. Honda, M., Hamazaki, T. S., Komazaki, S., Kagechika, H., Shudo, K., and Asashima, M. (2005) RXR agonist enhances the differentiation of cardiomyocytes derived from embryonic stem cells in serum-free conditions, *Biochem. Biophys. Res. Commun. 333*, 1334-1340.

79. Roberts, M. R., Hendrickson, A., McGuire, C. R., and Reh, T. A. (2005) Retinoid X receptor (gamma) is necessary to establish the S-opsin gradient in cone photoreceptors of the developing mouse retina, *Invest. Ophthalmol. Vis. Sci. 46*, 2897-2904.

80. Martin, D., Maestro, O., Cruz, J., Mane-Padros, D., and Belles, X. (2006) RNAi studies reveal a conserved role for RXR in molting in the cockroach Blattella germanica, *J Insect Physiol 52*, 410-416.

81. Zapata-Gonzalez, F., Rueda, F., Petriz, J., Domingo, P., Villarroya, F., de Madariaga, A., and Domingo, J. C. (2007) 9-cis-Retinoic acid (9cRA), a retinoid X receptor (RXR) ligand, exerts immunosuppressive effects on dendritic cells by RXR-dependent activation: inhibition of peroxisome proliferator-activated receptor gamma blocks some of the 9cRA activities, and precludes them to mature phenotype development, *J Immunol 178*, 6130-6139.

82. Siaussat, D., Bozzolan, F., Porcheron, P., and Debernard, S. (2007) Identification of steroid hormone signaling pathway in insect cell differentiation, *Cell. Mol. Life Sci. 64*, 365-376.

83. Taschner, S., Koesters, C., Platzer, B., Jorgl, A., Ellmeier, W., Benesch, T., and Strobl, H. (2007) Down-regulation of RXRalpha expression is essential for neutrophil development from granulocyte/monocyte progenitors, *Blood 109*, 971-979.

84. Rodriguez, A., Diez, C., Caamano, J. N., de Frutos, C., Royo, L. J., Munoz, M., Ikeda, S., Facal, N., Alvarez-Viejo, M., and Gomez, E. (2007) Retinoid receptor-specific agonists regulate bovine in vitro early embryonic development, differentiation and expression of genes related to cell cycle arrest and apoptosis, *Theriogenology 68*, 1118-1127.

85. Raisher, B. D., Gulick, T., Zhang, Z., Strauss, A. W., Moore, D. D., and Kelly, D. P. (1992) Identification of a novel retinoid-responsive element in the promoter region of the medium chain acyl-coenzyme A dehydrogenase gene, *J Biol Chem 267*, 20264-20269.

86. Lenhard, J. M., Lancaster, M. E., Paulik, M. A., Weiel, J. E., Binz, J. G., Sundseth, S. S., Gaskill, B. A., Lightfoot, R. M., and Brown, H. R. (1999) The RXR agonist LG100268 causes hepatomegaly, improves glycaemic control and decreases cardiovascular risk and cachexia in diabetic mice suffering from pancreatic beta-cell dysfunction, *Diabetologia 42*, 545-554.

87. Davies, P. J., Berry, S. A., Shipley, G. L., Eckel, R. H., Hennuyer, N., Crombie, D. L., Ogilvie, K. M., Peinado-Onsurbe, J., Fievet, C., Leibowitz, M. D., Heyman, R. A., and Auwerx, J. (2001) Metabolic effects of rexinoids: tissue-specific regulation of lipoprotein lipase activity, *Mol. Pharmacol. 59*, 170-176.

88. Singh Ahuja, H., Liu, S., Crombie, D. L., Boehm, M., Leibowitz, M. D., Heyman, R. A., Depre, C., Nagy, L., Tontonoz, P., and Davies, P. J. (2001) Differential effects of rexinoids and thiazolidinediones on metabolic gene expression in diabetic rodents, *Mol. Pharmacol. 59*, 765-773.

89. Vuligonda, V., Thacher, S. M., and Chandraratna, R. A. (2001) Enantioselective syntheses of potent retinoid X receptor ligands: differential biological activities of individual antipodes, *J Med Chem 44*, 2298-2303.

90. Canan Koch, S. S., Dardashti, L. J., Cesario, R. M., Croston, G. E., Boehm, M. F., Heyman, R. A., and Nadzan, A. M. (1999) Synthesis of retinoid X receptor-specific ligands that are potent inducers of adipogenesis in 3T3-L1 cells, *J Med Chem 42*, 742-750.

91. Metzger, D., Imai, T., Jiang, M., Takukawa, R., Desvergne, B., Wahli, W., and Chambon, P. (2005) Functional role of RXRs and PPARgamma in mature adipocytes, *Prostaglandins Leukot. Essent. Fatty Acids 73*, 51-58.

92. Norata, G. D., Ongari, M., Uboldi, P., Pellegatta, F., and Catapano, A. L. (2005) Liver X receptor and retinoic X receptor agonists modulate the expression of genes involved in lipid metabolism in human endothelial cells, *Int J Mol Med 16*, 717-722.

93. Schaiff, W. T., Bildirici, I., Cheong, M., Chern, P. L., Nelson, D. M., and Sadovsky, Y. (2005) Peroxisome proliferator-activated receptor-gamma and retinoid X receptor signaling regulate fatty acid uptake by primary human placental trophoblasts, *J Clin Endocrinol Metab 90*, 4267-4275.

94. Yotsumoto, T., Naitoh, T., Kanaki, T., and Tsuruzoe, N. (2005) A retinoid X

receptor antagonist, HX531, improves leptin resistance without increasing plasma leptin level in KK-Ay mice under normal dietary conditions, *Metabolism 54*, 573-578.

95. Grun, F., Watanabe, H., Zamanian, Z., Maeda, L., Arima, K., Cubacha, R., Gardiner, D. M., Kanno, J., Iguchi, T., and Blumberg, B. (2006) Endocrine-disrupting organotin compounds are potent inducers of adipogenesis in vertebrates, *Mol. Endocrinol. 20*, 2141-2155.

96. Gyamfi, M. A., Kocsis, M. G., He, L., Dai, G., Mendy, A. J., and Wan, Y. Y. (2006) The role of retinoid X receptor alpha in regulating alcohol metabolism, *J Pharmacol Exp Ther 319*, 360-368.

97. Ziouzenkova, O., Orasanu, G., Sharlach, M., Akiyama, T. E., Berger, J. P., Viereck, J., Hamilton, J. A., Tang, G., Dolnikowski, G. G., Vogel, S., Duester, G., and Plutzky, J. (2007) Retinaldehyde represses adipogenesis and diet-induced obesity, *Nat. Med. 13*, 695-702.

98. Yang, Y., Vacchio, M. S., and Ashwell, J. D. (1993) 9-cis-retinoic acid inhibits activation-driven T-cell apoptosis: implications for retinoid X receptor involvement in thymocyte development, *Proc Natl Acad Sci U S A 90*, 6170-6174.

99. Nicodeme, E., Nicaud, M., and Issandou, M. (1995) Retinoids stimulate fibrinogen production both in vitro (hepatocytes) and in vivo. Induction requires activation of the retinoid X receptor, *Arterioscler. Thromb. Vasc. Biol. 15*, 1660-1667.

100. Mukherjee, R., Davies, P. J., Crombie, D. L., Bischoff, E. D., Cesario, R. M., Jow, L., Hamann, L. G., Boehm, M. F., Mondon, C. E., Nadzan, A. M., Paterniti, J. R. J., and Heyman, R. A. (1997) Sensitization of diabetic and obese mice to insulin by retinoid X receptor agonists, *Nature 386*, 407-410.

101. Zelhof, A. C., Ghbeish, N., Tsai, C., Evans, R. M., and McKeown, M. (1997) A role for ultraspiracle, the Drosophila RXR, in morphogenetic furrow movement and photoreceptor cluster formation, *Development 124*, 2499-2506.

102. Duriez, P., and Fruchart, J. C. (1998) Post-statin approaches to hyperlipidaemia, *Expert Opin. Investig. Drugs 7*, 1997-2009.

103. Mascrez, B., Mark, M., Dierich, A., Ghyselinck, N. B., Kastner, P., and Chambon, P. (1998) The RXRalpha ligand-dependent activation function 2 (AF-2) is important for mouse development, *Development 125*, 4691-4707.

104. Shiohara, M., Dawson, M. I., Hobbs, P. D., Sawai, N., Higuchi, T., Koike, K., Komiyama, A., and Koeffler, H. P. (1999) Effects of novel RAR- and RXR-selective retinoids on myeloid leukemic proliferation and differentiation in vitro, *Blood 93*, 2057-2066.

105. Liu, Y. L., Sennitt, M. V., Hislop, D. C., Crombie, D. L., Heyman, R. A., and Cawthorne, M. A. (2000) Retinoid X receptor agonists have anti-obesity effects and improve insulin sensitivity in Zucker fa/fa rats, *Int J Obes Relat Metab Disord 24*, 997-1004.

106. Smit, J. V., Franssen, M. E. J., de Jong, E. M. G. J., Lambert, J., Roseeuw, D. I., De Weert, J., Yocum, R. C., Stevens, V. J., and van De Kerkhof, P. C. M. (2004) A phase II multicenter clinical trial of systemic bexarotene in psoriasis, *J. Am. Acad. Dermatol. 51*, 249-256.

107. Xu, J., Storer, P. D., Chavis, J. A., Racke, M. K., and Drew, P. D. (2005) Agonists for the peroxisome proliferator-activated receptor-alpha and the retinoid X receptor inhibit inflammatory responses of microglia, *J Neurosci Res 81*, 403-411.

108. Li, X., Hansen, P. A., Xi, L., Chandraratna, R. A. S., and Burant, C. F. (2005) Distinct mechanisms of glucose lowering by specific agonists for peroxisomal proliferator activated receptor gamma and retinoic acid X receptors, *J Biol Chem 280*, 38317-38327.

109. Deng, T., Shan, S., Li, Z., Wu, Z., Liao, C., Ko, B., Lu, X., Cheng, J., and Ning, Z. (2005) A new retinoid-like compound that activates peroxisome proliferator-activated receptors and lowers blood glucose in diabetic mice, *Biol. Pharm. Bull. 28*, 1192-1196.

110. Du, X., Tabeta, K., Mann, N., Crozat, K., Mudd, S., and Beutler, B. (2005) An essential role for Rxr alpha in the development of Th2 responses, *Eur J Immunol 35*, 3414-3423.

111. Rasooly, R., Schuster, G. U., Gregg, J. P., Xiao, J., Chandraratna, R. A. S., and Stephensen, C. B. (2005) Retinoid x receptor agonists increase bcl2a1 expression and decrease apoptosis of naive T lymphocytes, *J Immunol 175*, 7916-7929.

112. Li, M., Messaddeq, N., Teletin, M., Pasquali, J., Metzger, D., and Chambon, P. (2005) Retinoid X receptor ablation in adult mouse keratinocytes generates an atopic dermatitis triggered by thymic stromal lymphopoietin, *Proc Natl Acad Sci U S A 102*, 14795-14800.

113. Shaish, A., Harari, A., Hananshvili, L., Cohen, H., Bitzur, R., Luvish, T., Ulman, E., Golan, M., Ben-Amotz, A., Gavish, D., Rotstein, Z., and Harats, D. (2006) 9-cis beta-carotene-rich powder of the alga Dunaliella bardawil increases plasma HDL-cholesterol in fibrate-treated patients, *Atherosclerosis 189*, 215-221.

114. Lalloyer, F., Fievet, C., Lestavel, S., Torpier, G., van der Veen, J., Touche, V., Bultel, S., Yous, S., Kuipers, F., Paumelle, R., Fruchart, J., Staels, B., and Tailleux, A. (2006) The RXR agonist bexarotene improves cholesterol homeostasis and inhibits atherosclerosis progression in a mouse model of mixed dyslipidemia, *Arterioscler. Thromb. Vasc. Biol. 26*, 2731-2737.

115. Haraguchi, G., Suzuki, J., Kosuge, H., Ogawa, M., Koga, N., Muto, S., Itai, A., Kagechika, H., Shudo, K., and Isobe, M. (2006) A new RXR agonist, HX630, suppresses intimal hyperplasia in a mouse blood flow cessation model, *J Mol Cell Cardiol 41*, 885-892.

116. Xu, J., Chavis, J. A., Racke, M. K., and Drew, P. D. (2006) Peroxisome proliferator-activated receptor-alpha and retinoid X receptor agonists inhibit inflammatory responses of astrocytes, *J Neuroimmunol 176*, 95-105.

117. Leibowitz, M. D., Ardecky, R. J., Boehm, M. F., Broderick, C. L., Carfagna, M. A., Crombie, D. L., D'Arrigo, J., Etgen, G. J., Faul, M. M., Grese, T. A., Havel, H., Hein, N. I., Heyman, R. A., Jolley, D., Klausing, K., Liu, S., Mais, D. E., Mapes, C. M., Marschke, K. B., Michellys, P., Montrose-Rafizadeh, C., Ogilvie, K. M., Pascual, B., Rungta, D., Tyhonas, J. S., Urcan, M. S., Wardlow, M., Yumibe, N., and Reifel-Miller, A. (2006) Biological characterization of a heterodimer-selective retinoid X receptor modulator: potential benefits for the treatment of type 2 diabetes, *Endocrinology 147*, 1044-1053.

118. Grenningloh, R., Gho, A., di Lucia, P., Klaus, M., Bollag, W., Ho, I., Sinigaglia, F., and Panina-Bordignon, P. (2006) Cutting Edge: Inhibition of the retinoid X receptor (RXR) blocks T helper 2 differentiation and prevents allergic lung inflammation, *J Immunol 176*, 5161-5166.

119. Burrage, P. S., Huntington, J. T., Sporn, M. B., and Brinckerhoff, C. E. (2007) Regulation of matrix metalloproteinase gene expression by a retinoid X receptor-specific ligand, *Arthritis Rheum. 56*, 892-904.

120. Moraes, L. A., Swales, K. E., Wray, J. A., Damazo, A., Gibbins, J. M., Warner, T. D., and Bishop-Bailey, D. (2007) Nongenomic signaling of the retinoid X receptor through binding and inhibiting Gq in human platelets, *Blood 109*, 3741-3744.

121. Stephensen, C. B., Borowsky, A. D., and Lloyd, K. C. K. (2007) Disruption of Rxra gene in thymocytes and T lymphocytes modestly alters lymphocyte frequencies, proliferation, survival and T helper type 1/type 2 balance, *Immunology 121*, 484-498.

122. Sakaki, J., Kishida, M., Konishi, K., Gunji, H., Toyao, A., Matsumoto, Y., Kanazawa, T., Uchiyama, H., Fukaya, H., Mitani, H., Arai, Y., and Kimura, M. (2007) Synthesis and structure-activity relationship of novel RXR antagonists: orally active anti-diabetic and anti-obesity agents, *Bioorg. Med. Chem. Lett. 17*, 4804-4807.

123. Wietrzych, M., Meziane, H., Sutter, A., Ghyselinck, N., Chapman, P. F., Chambon, P., and Krezel, W. (2005) Working memory deficits in retinoid X receptor gamma-deficient mice, *Learn Mem. 12*, 318-326.

124. Brown, N. S., Smart, A., Sharma, V., Brinkmeier, M. L., Greenlee, L.,

Camper, S. A., Jensen, D. R., Eckel, R. H., Krezel, W., Chambon, P., and Haugen, B. R. (2000) Thyroid hormone resistance and increased metabolic rate in the RXR-gamma-deficient mouse, *J Clin Invest 106*, 73-79.

125. Haugen, B. R., Jensen, D. R., Sharma, V., Pulawa, L. K., Hays, W. R., Krezel, W., Chambon, P., and Eckel, R. H. (2004) Retinoid X receptor gamma-deficient mice have increased skeletal muscle lipoprotein lipase activity and less weight gain when fed a high-fat diet, *Endocrinology 145*, 3679-3685.

126. Lotan, R., Dawson, M. I., Zou, C. C., Jong, L., Lotan, D., and Zou, C. P. (1995) Enhanced efficacy of combinations of retinoic acid- and retinoid X receptor-selective retinoids and alpha-interferon in inhibition of cervical carcinoma cell proliferation, *Cancer Res. 55*, 232-236.

127. Spanjaard, R. A., Sugawara, A., Ikeda, M., and Chin, W. W. (1995) Evidence that retinoid X receptors mediate retinoid-dependent transcriptional activation of the retinoic acid receptor beta gene in S91 melanoma cells, *J Biol Chem 270*, 17429-17436.

128. Boehm, M. F., Zhang, L., Zhi, L., McClurg, M. R., Berger, E., Wagoner, M., Mais, D. E., Suto, C. M., Davies, J. A., and Heyman, R. A. (1995) Design and synthesis of potent retinoid X receptor selective ligands that induce apoptosis in leukemia cells, *J Med Chem 38*, 3146-3155.

129. Horn, V., Minucci, S., Ogryzko, V. V., Adamson, E. D., Howard, B. H., Levin, A. A., and Ozato, K. (1996) RAR and RXR selective ligands cooperatively induce apoptosis and neuronal differentiation in P19 embryonal carcinoma cells, *FASEB J. 10*, 1071-1077.

130. Gottardis, M. M., Bischoff, E. D., Shirley, M. A., Wagoner, M. A., Lamph, W. W., and Heyman, R. A. (1996) Chemoprevention of mammary carcinoma by LGD1069 (Targretin): an RXR-selective ligand, *Cancer Res. 56*, 5566-5570.

131. Miller, V. A., Benedetti, F. M., Rigas, J. R., Verret, A. L., Pfister, D. G., Straus, D., Kris, M. G., Crisp, M., Heyman, R., Loewen, G. R., Truglia, J. A., and Warrell, R. P. J. (1997) Initial clinical trial of a selective retinoid X receptor ligand, LGD1069, *J Clin Oncol 15*, 790-795.

132. de Vos, S., Dawson, M. I., Holden, S., Le, T., Wang, A., Cho, S. K., Chen, D. L., and Koeffler, H. P. (1997) Effects of retinoid X receptor-selective ligands on proliferation of prostate cancer cells, *Prostate 32*, 115-121.

133. Wu, Q., Dawson, M. I., Zheng, Y., Hobbs, P. D., Agadir, A., Jong, L., Li, Y., Liu, R., Lin, B., and Zhang, X. K. (1997) Inhibition of trans-retinoic acid-resistant human breast cancer cell growth by retinoid X receptor-selective retinoids, *Mol. Cell. Biol. 17*, 6598-6608.

134. Sun, S. Y., Yue, P., Dawson, M. I., Shroot, B., Michel, S., Lamph, W. W., Heyman, R. A., Teng, M., Chandraratna, R. A., Shudo, K., Hong, W. K., and Lotan, R. (1997) Differential effects of synthetic nuclear retinoid receptor-selective retinoids on the growth of human non-small cell lung carcinoma cells, *Cancer Res. 57*, 4931-4939.

135. Bischoff, E. D., Gottardis, M. M., Moon, T. E., Heyman, R. A., and Lamph, W. W. (1998) Beyond tamoxifen: the retinoid X receptor-selective ligand LGD1069 (TARGRETIN) causes complete regression of mammary carcinoma, *Cancer Res. 58*, 479-484.

136. Wan, H., Dawson, M. I., Hong, W. K., and Lotan, R. (1998) Overexpressed activated retinoid X receptors can mediate growth inhibitory effects of retinoids in human carcinoma cells, *J Biol Chem 273*, 26915-26922.

137. Rosati, R., Ramnath, N., Adil, M. R., Ou, X., Ali, M. A., Heyman, R. A., and Kalemkerian, G. P. (1998) Activity of 9-cis-retinoic acid and receptor-selective retinoids in small cell lung cancer cell lines, *Anticancer Res. 18*, 4071-4075.

138. Rizvi, N. A., Marshall, J. L., Dahut, W., Ness, E., Truglia, J. A., Loewen, G., Gill, G. M., Ulm, E. H., Geiser, R., Jaunakais, D., and Hawkins, M. J. (1999) A Phase I study of LGD1069 in adults with advanced cancer, *Clin. Cancer Res. 5*, 1658-1664.

139. Pakala, R., and Benedict, C. R. (1999) Modulation of endothelial cell proliferation by retinoid x receptor agonists, *Eur J Pharmacol 385*, 255-261.

140. Bischoff, E. D., Heyman, R. A., and Lamph, W. W. (1999) Effect of the retinoid X receptor-selective ligand LGD1069 on mammary carcinoma after tamoxifen failure, *J Natl Cancer Inst 91*, 2118.

141. Liu, B., Lee, K., Li, H., Ma, L., Lin, G. L., Chandraratna, R. A. S., and Cohen, P. (2005) Combination therapy of insulin-like growth factor binding protein-3 and retinoid X receptor ligands synergize on prostate cancer cell apoptosis in vitro and in vivo, *Clin. Cancer Res. 11*, 4851-4856.

142. Altucci, L., Rossin, A., Hirsch, O., Nebbioso, A., Vitoux, D., Wilhelm, E., Guidez, F., De Simone, M., Schiavone, E. M., Grimwade, D., Zelent, A., de The, H., and Gronemeyer, H. (2005) Rexinoid-triggered differentiation and tumor-selective apoptosis of acute myeloid leukemia by protein kinase A-mediated desubordination of retinoid X receptor, *Cancer Res. 65*, 8754-8765.

143. Edelman, M. J., Smith, R., Hausner, P., Doyle, L. A., Kalra, K., Kendall, J., Bedor, M., and Bisaccia, S. (2005) Phase II trial of the novel retinoid, bexarotene, and gemcitabine plus carboplatin in advanced non-small-cell lung cancer, *J Clin Oncol 23*, 5774-5778.

144. Foss, F., Demierre, M. F., and DiVenuti, G. (2005) A phase-1 trial of bexarotene and denileukin diftitox in patients with relapsed or refractory cutaneous T-cell lymphoma, *Blood 106*, 454-457.

145. Wang, Y., Zhang, Z., Yao, R., Jia, D., Wang, D., Lubet, R. A., and You, M. (2006) Prevention of lung cancer progression by bexarotene in mouse models, *Oncogene 25*, 1320-1329.

146. Gamage, S. D., Bischoff, E. D., Burroughs, K. D., Lamph, W. W., Gottardis, M. M., Walker, C. L., and Fuchs-Young, R. (2000) Efficacy of LGD1069 (Targretin), a retinoid X receptor-selective ligand, for treatment of uterine leiomyoma, *J Pharmacol Exp Ther 295*, 677-681.

147. Agarwal, V. R., Bischoff, E. D., Hermann, T., and Lamph, W. W. (2000) Induction of adipocyte-specific gene expression is correlated with mammary tumor regression by the retinoid X receptor-ligand LGD1069 (targretin), *Cancer Res. 60*, 6033-6038.

148. Rizvi, N., Hawkins, M. J., Eisenberg, P. D., Yocum, R. C., and Reich, S. D. (2001) Placebo-controlled trial of bexarotene, a retinoid x receptor agonist, as maintenance therapy for patients treated with chemotherapy for advanced non-small-cell lung cancer, *Clin. Lung Cancer 2*, 210-215.

149. Duvic, M., Hymes, K., Heald, P., Breneman, D., Martin, A. G., Myskowski, P., Crowley, C., and Yocum, R. C. (2001) Bexarotene is effective and safe for treatment of refractory advanced-stage cutaneous T-cell lymphoma: multinational phase II-III trial results, *J Clin Oncol 19*, 2456-2471.

150. Khuri, F. R., Rigas, J. R., Figlin, R. A., Gralla, R. J., Shin, D. M., Munden, R., Fox, N., Huyghe, M. R., Kean, Y., Reich, S. D., and Hong, W. K. (2001) Multi-institutional phase I/II trial of oral bexarotene in combination with cisplatin and vinorelbine in previously untreated patients with advanced non-small-cell lung cancer, *J Clin Oncol 19*, 2626-2637.

151. Wu, K., Kim, H., Rodriquez, J. L., Hilsenbeck, S. G., Mohsin, S. K., Xu, X., Lamph, W. W., Kuhn, J. G., Green, J. E., and Brown, P. H. (2002) Suppression of mammary tumorigenesis in transgenic mice by the RXR-selective retinoid, LGD1069, *Cancer Epidemiol. Biomarkers Prev. 11*, 467-474.

152. Wu, K., Zhang, Y., Xu, X., Hill, J., Celestino, J., Kim, H., Mohsin, S. K., Hilsenbeck, S. G., Lamph, W. W., Bissonette, R., and Brown, P. H. (2002) The retinoid X receptor-selective retinoid, LGD1069, prevents the development of estrogen receptor-negative mammary tumors in transgenic mice, *Cancer Res. 62*, 6376-6380.

153. Krathen, R. A., Ward, S., and Duvic, M. (2003) Bexarotene is a new treatment option for lymphomatoid papulosis, *Dermatology 206*, 142-

147.

154. Crowe, D. L., and Chandraratna, R. A. S. (2004) A retinoid X receptor (RXR)-selective retinoid reveals that RXR-alpha is potentially a therapeutic target in breast cancer cell lines, and that it potentiates antiproliferative and apoptotic responses to peroxisome proliferator-activated receptor ligands, *Breast Cancer Res. 6*, R546-55.

155. Balasubramanian, S., Chandraratna, R. A. S., and Eckert, R. L. (2004) Suppression of human pancreatic cancer cell proliferation by AGN194204, an RXR-selective retinoid, *Carcinogenesis 25*, 1377-1385.

156. Yen, W., Corpuz, M. R., Prudente, R. Y., Cooke, T. A., Bissonnette, R. P., Negro-Vilar, A., and Lamph, W. W. (2004) A selective retinoid X receptor agonist bexarotene (Targretin) prevents and overcomes acquired paclitaxel (Taxol) resistance in human non-small cell lung cancer, *Clin. Cancer Res. 10*, 8656-8664.

157. Kolluri, S. K., Corr, M., James, S. Y., Bernasconi, M., Lu, D., Liu, W., Cottam, H. B., Leoni, L. M., Carson, D. A., and Zhang, X. (2005) The R-enantiomer of the nonsteroidal antiinflammatory drug etodolac binds retinoid X receptor and induces tumor-selective apoptosis, *Proc Natl Acad Sci U S A 102*, 2525-2530.

158. Yen, W., Prudente, R. Y., Corpuz, M. R., Negro-Vilar, A., and Lamph, W. W. (2006) A selective retinoid X receptor agonist bexarotene (LGD1069, targretin) inhibits angiogenesis and metastasis in solid tumours, *Br. J. Cancer 94*, 654-660.

159. Yen, W., and Lamph, W. W. (2006) A selective retinoid X receptor agonist bexarotene (LGD1069, Targretin) prevents and overcomes multidrug resistance in advanced prostate cancer, *Prostate 66*, 305-316.

160. Zeng, J., Sun, D., Wang, L., Cao, X., Qi, J., Yang, T., Hu, C., Liu, W., and Zhang, X. (2006) Hypericum sampsonii induces apoptosis and nuclear export of retinoid X receptor-alpha, *Carcinogenesis 27*, 1991-2000.

161. Zhu, J., Nasr, R., Peres, L., Riaucoux-Lormiere, F., Honore, N., Berthier, C., Kamashev, D., Zhou, J., Vitoux, D., Lavau, C., and de, T. H. (2007) RXR Is an Essential Component of the Oncogenic PML/RARA Complex In Vivo, *Cancer Cell 12*, 23-35.

162. Dragnev, K. H., Petty, W. J., Shah, S. J., Lewis, L. D., Black, C. C., Memoli, V., Nugent, W. C., Hermann, T., Negro-Vilar, A., Rigas, J. R., and Dmitrovsky, E. (2007) A proof-of-principle clinical trial of bexarotene in patients with non-small cell lung cancer, *Clin. Cancer Res. 13*, 1794-1800.

163. Fu, J., Ding, Y., Huang, D., Li, H., and Chen, X. (2007) The retinoid X

receptor-selective ligand, LGD1069, inhibits tumor-induced angiogenesis via suppression of VEGF in human non-small cell lung cancer, *Cancer Lett. 248*, 153-163.

164. Tooker, P., Yen, W., Ng, S., Negro-Vilar, A., and Hermann, T. W. (2007) Bexarotene (LGD1069, Targretin), a selective retinoid X receptor agonist, prevents and reverses gemcitabine resistance in NSCLC cells by modulating gene amplification, *Cancer Res. 67*, 4425-4433.

165. Indra, A. K., Castaneda, E., Antal, M. C., Jiang, M., Messaddeq, N., Meng, X., Loehr, C. V., Gariglio, P., Kato, S., Wahli, W., Desvergne, B., Metzger, D., and Chambon, P. (2007) Malignant transformation of DMBA/TPA-induced papillomas and nevi in the skin of mice selectively lacking retinoid-X-receptor alpha in epidermal keratinocytes, *J Invest Dermatol 127*, 1250-1260.

166. Kanamori, T., Shimizu, M., Okuno, M., Matsushima-Nishiwaki, R., Tsurumi, H., Kojima, S., and Moriwaki, H. (2007) Synergistic growth inhibition by acyclic retinoid and vitamin K2 in human hepatocellular carcinoma cells, *Cancer Sci. 98*, 431-437.

167. Lattuada, D., Vigano, P., Mangioni, S., Sassone, J., Di Francesco, S., Vignali, M., and Di Blasio, A. M. (2007) Accumulation of retinoid X receptor-alpha in uterine leiomyomas is associated with a delayed ligand-dependent proteasome-mediated degradation and an alteration of its transcriptional activity, *Mol. Endocrinol. 21*, 602-612.

168. Tang, X., Suh, M., Li, R., and Gudas, L. J. (2007) Cell proliferation inhibition and alterations in retinol esterification induced by phytanic acid and docosahexaenoic acid, *J Lipid Res 48*, 165-176.

169. Huan, B., and Siddiqui, A. (1992) Retinoid X receptor RXR alpha binds to and trans-activates the hepatitis B virus enhancer, *Proc Natl Acad Sci U S A 89*, 9059-9063.

170. Huan, B., and Siddiqui, A. (1993) Regulation of hepatitis B virus gene expression, *J Hepatol 17 Suppl 3*, S20-3.

171. Sista, N. D., Pagano, J. S., Liao, W., and Kenney, S. (1993) Retinoic acid is a negative regulator of the Epstein-Barr virus protein (BZLF1) that mediates disruption of latent infection, *Proc Natl Acad Sci U S A 90*, 3894-3898.

172. Garcia, A. D., Ostapchuk, P., and Hearing, P. (1993) Functional interaction of nuclear factors EF-C, HNF-4, and RXR alpha with hepatitis B virus enhancer I, *J Virol 67*, 3940-3950.

173. Sista, N. D., Barry, C., Sampson, K., and Pagano, J. (1995) Physical and functional interaction of the Epstein-Barr virus BZLF1 transactivator with the retinoic acid receptors RAR alpha and RXR alpha, *Nucleic Acids Res. 23*, 1729-1736.

174. Kong, H. J., Hong, S. H., Lee, M. Y., Kim, H. D., Lee, J. W., and Cheong, J. (2000) Direct binding of hepatitis B virus X protein and retinoid X receptor contributes to phosphoenolpyruvate carboxykinase gene transactivation, *FEBS Lett. 483*, 114-118.

175. Tsutsumi, T., Suzuki, T., Shimoike, T., Suzuki, R., Moriya, K., Shintani, Y., Fujie, H., Matsuura, Y., Koike, K., and Miyamura, T. (2002) Interaction of hepatitis C virus core protein with retinoid X receptor alpha modulates its transcriptional activity, *Hepatology 35*, 937-946.

176. Levin, A. A., Sturzenbecker, L. J., Kazmer, S., Bosakowski, T., Huselton, C., Allenby, G., Speck, J., Kratzeisen, C., Rosenberger, M., Lovey, A., and et al (1992) 9-cis retinoic acid stereoisomer binds and activates the nuclear receptor RXR alpha, *Nature 355*, 359-361.

177. Heyman, R. A., Mangelsdorf, D. J., Dyck, J. A., Stein, R. B., Eichele, G., Evans, R. M., and Thaller, C. (1992) 9-cis retinoic acid is a high affinity ligand for the retinoid X receptor, *Cell 68*, 397-406.

178. Kitareewan, S., Burka, L. T., Tomer, K. B., Parker, C. E., Deterding, L. J., Stevens, R. D., Forman, B. M., Mais, D. E., Heyman, R. A., McMorris, T., and Weinberger, C. (1996) Phytol metabolites are circulating dietary factors that activate the nuclear receptor RXR, *Mol. Biol. Cell 7*, 1153-1166.

179. Mic, F. A., Molotkov, A., Benbrook, D. M., and Duester, G. (2003) Retinoid activation of retinoic acid receptor but not retinoid X receptor is sufficient to rescue lethal defect in retinoic acid synthesis, *Proc Natl Acad Sci USA 100*, 7135-7140.

180. Calleja, C., Messaddeq, N., Chapellier, B., Yang, H., Krezel, W., Li, M., Metzger, D., Mascrez, B., Ohta, K., Kagechika, H., Endo, Y., Mark, M., Ghyselinck, N. B., and Chambon, P. (2006) Genetic and pharmacological evidence that a retinoic acid cannot be the RXR-activating ligand in mouse epidermis keratinocytes, *Genes Dev. 20*, 1525-1538.

181. Wolf, G. (2006) Is 9-cis-retinoic acid the endogenous ligand for the retinoic acid-X receptor?, *Nutr. Rev. 64*, 532-538.

182. Eager, N. S., Brickell, P. M., Snell, C., and Wood, J. N. (1992) Structural and functional evidence for activation of a chick retinoid X receptor by eicosanoids, *Proc. Biol. Sci. 250*, 63-69.

183. Lemotte, P. K., Keidel, S., and Apfel, C. M. (1996) Phytanic acid is a retinoid X receptor ligand, *Eur J Biochem 236*, 328-333.

184. de Urquiza, A. M., Liu, S., Sjoberg, M., Zetterstrom, R. H., Griffiths, W., Sjovall, J., and Perlmann, T. (2000) Docosahexaenoic acid, a ligand for the retinoid X receptor in mouse brain, *Science 290*, 2140-2144.

185. Radominska-Pandya, A., and Chen, G. (2002) Photoaffinity labeling of human retinoid X receptor beta (RXRbeta) with 9-cis-retinoic acid: identification of phytanic acid, docosahexaenoic acid, and lithocholic acid as ligands for RXRbeta, *Biochemistry 41*, 4883-4890.

186. Goldstein, J. T., Dobrzyn, A., Clagett-Dame, M., Pike, J. W., and DeLuca, H. F. (2003) Isolation and characterization of unsaturated fatty acids as natural ligands for the retinoid-X receptor, *Arch. Biochem. Biophys. 420*, 185-193.

187. Lengqvist, J., Mata De Urquiza, A., Bergman, A., Willson, T. M., Sjovall, J., Perlmann, T., and Griffiths, W. J. (2004) Polyunsaturated fatty acids including docosahexaenoic and arachidonic acid bind to the retinoid X receptor alpha ligand-binding domain, *Mol. Cell. Proteomics. 3*, 692-703.

188. Fischer, H., Dias, S. M., Santos, M. A., Alves, A. C., Zanchin, N., Craievich, A. F., Apriletti, J. W., Baxter, J. D., Webb, P., Neves, F. A., Ribeiro, R. C., and Polikarpov, I. (2003) Low resolution structures of the retinoid X receptor DNA-binding and ligand-binding domains revealed by synchrotron X-ray solution scattering, *J Biol Chem 278*, 16030-16038.

189. Bourguet, W., Ruff, M., Chambon, P., Gronemeyer, H., and Moras, D. (1995) Crystal structure of the ligand-binding domain of the human nuclear receptor RXR-alpha, *Nature 375*, 377-382.

190. Egea, P. F., Mitschler, A., Rochel, N., Ruff, M., Chambon, P., and Moras, D. (2000) Crystal structure of the human RXRalpha ligand-binding domain bound to its natural ligand: 9-cis retinoic acid, *EMBO J. 19*, 2592-2601.

191. Gampe, R T, Jr, Montana, V. G., Lambert, M. H., Wisely, G. B., Milburn, M. V., and Xu, H. E. (2000) Structural basis for autorepression of retinoid X receptor by tetramer formation and the AF-2 helix, *Genes Dev. 14*, 2229-2241.

192. Egea, P. F., Mitschler, A., and Moras, D. (2002) Molecular recognition of agonist ligands by RXRs, *Mol. Endocrinol. 16*, 987-997.

193. de Groot, A., de Rosny, E., Juillan-Binard, C., Ferrer, J. L., Laudet, V., Pierce, R. J., Pebay-Peyroula, E., Fontecilla-Camps, J. C., and Borel, F. (2005) Crystal structure of a novel tetrameric complex of agonist-bound ligand-binding domain of Biomphalaria glabrata retinoid X receptor, *J Mol Biol 354*, 841-853.

194. Bourguet, W., Andry, V., Iltis, C., Klaholz, B., Potier, N., Van Dorsselaer, A., Chambon, P., Gronemeyer, H., and Moras, D. (2000) Heterodimeric complex of RAR and RXR nuclear receptor ligand-binding domains: purification, crystallization, and preliminary X-ray diffraction analysis, *Protein Expr. Purif. 19*, 284-288.

195. Gampe, R. T. J., Montana, V. G., Lambert, M. H., Miller, A. B., Bledsoe, R. K., Milburn, M. V., Kliewer, S. A., Willson, T. M., and Xu, H. E. (2000) Asymmetry in the PPARgamma/RXRalpha crystal structure reveals the molecular basis of heterodimerization among nuclear receptors, *Mol. Cell 5*, 545-555.

196. Xu, H. E., Lambert, M. H., Montana, V. G., Plunket, K. D., Moore, L. B., Collins, J. L., Oplinger, J. A., Kliewer, S. A., Gampe, R. T. J., McKee, D. D., Moore, J. T., and Willson, T. M. (2001) Structural determinants of ligand binding selectivity between the peroxisome proliferator-activated receptors, *Proc Natl Acad Sci U S A 98*, 13919-13924.

197. Suino, K., Peng, L., Reynolds, R., Li, Y., Cha, J. Y., Repa, J. J., Kliewer, S. A., and Xu, H. E. (2004) The nuclear xenobiotic receptor CAR: structural determinants of constitutive activation and heterodimerization, *Mol. Cell 16*, 893-905.

198. Xu, R. X., Lambert, M. H., Wisely, B. B., Warren, E. N., Weinert, E. E., Waitt, G. M., Williams, J. D., Collins, J. L., Moore, L. B., Willson, T. M., and Moore, J. T. (2004) A structural basis for constitutive activity in the human CAR/RXRalpha heterodimer, *Mol. Cell 16*, 919-928.

199. Haffner, C. D., Lenhard, J. M., Miller, A. B., McDougald, D. L., Dwornik, K., Ittoop, O. R., Gampe, R T, Jr, Xu, H. E., Blanchard, S., Montana, V. G., Consler, T. G., Bledsoe, R. K., Ayscue, A., and Croom, D. (2004) Structure-based design of potent retinoid X receptor alpha agonists, *J Med Chem 47*, 2010-2029.

200. Pogenberg, V., Guichou, J. F., Vivat-Hannah, V., Kammerer, S., Perez, E., Germain, P., de Lera, A. R., Gronemeyer, H., Royer, C. A., and Bourguet, W. (2005) Characterization of the interaction between retinoic acid receptor/retinoid X receptor (RAR/RXR) heterodimers and transcriptional coactivators through structural and fluorescence anisotropy studies, *J Biol Chem 280*, 1625-1633.

201. Lu, J., Cistola, D. P., and Li, E. (2006) Analysis of ligand binding and protein dynamics of human retinoid X receptor alpha ligand-binding domain by nuclear magnetic resonance, *Biochemistry 45*, 1629-1639.

202. Love, J. D., Gooch, J. T., Benko, S., Li, C., Nagy, L., Chatterjee, V. K. K., Evans, R. M., and Schwabe, J. W. R. (2002) The structural basis for the specificity of retinoid-X receptor-selective agonists: new insights into the role of helix H12, *J Biol Chem 277*, 11385-11391.

203. Svensson, S., Ostberg, T., Jacobsson, M., Norstrom, C., Stefansson, K., Hallen, D., Johansson, I. C., Zachrisson, K., Ogg, D., and Jendeberg, L. (2003) Crystal structure of the heterodimeric complex of LXRalpha and RXRbeta ligand-binding domains in a fully agonistic conformation, *EMBO J. 22*, 4625-4633.

204. Jaye, M. C., Krawiec, J. A., Campobasso, N., Smallwood, A., Qiu, C., Lu,

Q., Kerrigan, J. J., De Los Frailes Alvaro, M., Laffitte, B., Liu, W., Marino, J. P. J., Meyer, C. R., Nichols, J. A., Parks, D. J., Perez, P., Sarov-Blat, L., Seepersaud, S. D., Steplewski, K. M., Thompson, S. K., Wang, P., Watson, M. A., Webb, C. L., Haigh, D., Caravella, J. A., Macphee, C. H., Willson, T. M., and Collins, J. L. (2005) Discovery of substituted maleimides as liver X receptor agonists and determination of a ligand-bound crystal structure, *J Med Chem 48*, 5419-5422.

205. Billas, I. M., Moulinier, L., Rochel, N., and Moras, D. (2001) Crystal structure of the ligand-binding domain of the ultraspiracle protein USP, the ortholog of retinoid X receptors in insects, *J Biol Chem 276*, 7465-7474.

206. Clayton, G. M., Peak-Chew, S. Y., Evans, R. M., and Schwabe, J. W. (2001) The structure of the ultraspiracle ligand-binding domain reveals a nuclear receptor locked in an inactive conformation, *Proc Natl Acad Sci U S A 98*, 1549-1554.

207. Billas, I. M., Iwema, T., Garnier, J. M., Mitschler, A., Rochel, N., and Moras, D. (2003) Structural adaptability in the ligand-binding pocket of the ecdysone hormone receptor, *Nature 426*, 91-96.

208. Carmichael, J. A., Lawrence, M. C., Graham, L. D., Pilling, P. A., Epa, V. C., Noyce, L., Lovrecz, G., Winkler, D. A., Pawlak-Skrzecz, A., Eaton, R. E., Hannan, G. N., and Hill, R. J. (2005) The X-ray structure of a hemipteran ecdysone receptor ligand-binding domain: comparison with a lepidopteran ecdysone receptor ligand-binding domain and implications for insecticide design, *J Biol Chem 280*, 22258-22269.

209. Iwema, T., Billas, I. M. L., Beck, Y., Bonneton, F., Nierengarten, H., Chaumot, A., Richards, G., Laudet, V., and Moras, D. (2007) Structural and functional characterization of a novel type of ligand-independent RXR-USP receptor, *EMBO J. 26*, 3770-3782.

210. Lee, M. S., Kliewer, S. A., Provencal, J., Wright, P. E., and Evans, R. M. (1993) Structure of the retinoid X receptor alpha DNA binding domain: a helix required for homodimeric DNA binding, *Science 260*, 1117-1121.

211. Lee, M. S., Sem, D. S., Kliewer, S. A., Provencal, J., Evans, R. M., and Wright, P. E. (1994) NMR assignments and secondary structure of the retinoid X receptor alpha DNA-binding domain. Evidence for the novel C-terminal helix, *Eur J Biochem 224*, 639-650.

212. Holmbeck, S. M., Dyson, H. J., and Wright, P. E. (1998) DNA-induced conformational changes are the basis for cooperative dimerization by the DNA binding domain of the retinoid X receptor, *J Mol Biol 284*, 533-539.

213. Holmbeck, S. M., Foster, M. P., Casimiro, D. R., Sem, D. S., Dyson, H. J., and Wright, P. E. (1998) High-resolution solution structure of the

retinoid X receptor DNA-binding domain, *J Mol Biol 281*, 271-284.

214. van Tilborg, P. J., Czisch, M., Mulder, F. A., Folkers, G. E., Bonvin, A. M., Nair, M., Boelens, R., and Kaptein, R. (2000) Changes in dynamical behavior of the retinoid X receptor DNA-binding domain upon binding to a 14 base-pair DNA half site, *Biochemistry 39*, 8747-8757.

215. Biggins, J. B., and Koh, J. T. (2007) Chemical biology of steroid and nuclear hormone receptors, *Curr. Opin. Chem. Biol. 11*, 99-110.

216. Peet, D. J., Doyle, D. F., Corey, D. R., and Mangelsdorf, D. J. (1998) Engineering novel specificities for ligand-activated transcription in the nuclear hormone receptor RXR, *Chem Biol 5*, 13-21.

217. Doyle, D. F., Braasch, D. A., Jackson, L. K., Weiss, H. E., Boehm, M. F., Mangelsdorf, D. J., and Corey, D. R. (2001) Engineering orthogonal ligand-receptor pairs from "near drugs", *J Am Chem Soc 123*, 11367-11371.

218. Azizi, B., Chang, E. I., and Doyle, D. F. (2003) Chemical complementation: small-molecule-based genetic selection in yeast, *Biochem. Biophys. Res. Commun. 306*, 774-780.

219. Schwimmer, L. J., Rohatgi, P., Azizi, B., Seley, K. L., and Doyle, D. F. (2004) Creation and discovery of ligand-receptor pairs for transcriptional control with small molecules, *Proc Natl Acad Sci U S A 101*, 14707-14712.

220. Azizi, B. (2005) Chemical complementation: a genetic selection system for drug discovery, protein engineering and deciphering biosynthetic pathways. Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA.

221. Schwimmer, L. J. (2005) Engineering ligand-receptor pairs for small molecule control of transcription. Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA.

222. Belshaw, P. J., Schoepfer, J. G., Liu, K. Q., Morrison, K. L., and Schreiber, S. L. (1995) Rational design of orthogonal receptor-ligand combinations, *Angew. Chem. Int. Ed. Engl. 34*, 2132.

223. Matys, V., Fricke, E., Geffers, R., Gößling, E., Haubrock, M., Hehl, R., Hornischer, K., Karas, D., Kel, A. E., Kel-Margoulis, O. V., Kloos, D. U., Land, S., Lewicki-Potapov, B., Michael, H., Munch, R., Reuter, I., Rotert, S., Saxel, H., Scheer, M., Thiele, S., and Wingender, E. (2003) TRANSFAC: transcriptional regulation, from patterns to profiles, *Nucleic Acids Res. 31*, 374-378.

224. Altschul, S. F., Gish, W., Miller, W., Meyers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool, *J Mol Biol 215*, 403-410.

225. Falquet, L., Pagni, M., Bucher, P., Hulo, N., Sigrist, C. J. A., Hofmann, K., and Bairoch, A. (2002) The PROSITE database, its status in 2002, *Nucleic Acids Res. 30*, 235-238.

226. Rice, P., Longden, I., and Bleasby, A. (2000) EMBOSS: the European Molecular Biology Open Software Suite, *Trends Genet. 16*, 276-277.

227. Grillo, G., Licciulli, F., Liuni, S., Sbisa, E., and Pesole, G. (2003) PatSearch: A program for the detection of patterns and structural motifs in nucleotide sequences, *Nucleic Acids Res. 31*, 3608-3612.

228. Vousden, K. H., and Lu, X. (2002) Live or let die: the cell's response to p53, *Nat. Rev. Cancer 2*, 594-604.

229. Beerli, R. R., and Barbas, C F, 3rd (2002) Engineering polydactyl zinc-finger transcription factors, *Nat. Biotechnol. 20*, 135-141.

230. Jantz, D., Amann, B. T., Gatto, G J, Jr, and Berg, J. M. (2004) The design of functional DNA-binding proteins based on zinc finger domains, *Chem Rev 104*, 789-800.

231. Falke, D., Fisher, M., Ye, D., and Juliano, R. L. (2003) Design of artificial transcription factors to selectively regulate the pro-apoptotic bax gene, *Nucleic Acids Res. 31*, e10.

232. Bae, K. H., Do Kwon, Y., Shin, H. C., Hwang, M. S., Ryu, E. H., Park, K. S., Yang, H. Y., Lee, D. K., Lee, Y., Park, J., Sun Kwon, H., Kim, H. W., Yeh, B. I., Lee, H. W., Hyung Sohn, S., Yoon, J., Seol, W., and Kim, J. S. (2003) Human zinc fingers as building blocks in the construction of artificial transcription factors, *Nat. Biotechnol. 21*, 275-280.

233. Ptashne, M. (1992) *A genetic switch* 2nd Ed., Blackwell Scientific Publications and Cell Press, Cambridge, MA, .

234. Kohler, J. J., Metallo, S. J., Schneider, T. L., and Schepartz, A. (1999) DNA specificity enhanced by sequential binding of protein monomers, *Proc Natl Acad Sci USA 96*, 11735-11739.

235. Kohler, J. J., and Schepartz, A. (2001) Effects of nucleic acids and polyanions on dimer formation and DNA binding by bZIP and bHLHZip transcription factors, *Bioorg. Med. Chem. 9*, 2435-2443.

236. Wolfe, S. A., Grant, R. A., and Pabo, C. O. (2003) Structure of a designed dimeric zinc finger protein bound to DNA, *Biochemistry 42*, 13401-13409.

237. Fortin, A., Cregan, S. P., MacLaurin, J. G., Kushwaha, N., Hickman, E. S., Thompson, C. S., Hakim, A., Albert, P. R., Cecconi, F., Helin, K., Park, D. S., and Slack, R. S. (2001) APAF1 is a key transcriptional target for p53 in the regulation of neuronal cell death, *J Cell Biol 155*, 207-216.

238. Miyashita, T., and Reed, J. C. (1995) Tumor suppressor p53 is a direct transcriptional activator of the human bax gene, *Cell 80*, 293-299.

239. Bourdon, J. C., Deguin-Chambon, V., Lelong, J. C., Dessen, P., May, P., Debuire, B., and May, E. (1997) Further characterisation of the p53 responsive element--identification of new candidate genes for trans-activation by p53, *Oncogene 14*, 85-94.

240. Oda, K., Arakawa, H., Tanaka, T., Matsuda, K., Tanikawa, C., Mori, T., Nishimori, H., Tamai, K., Tokino, T., Nakamura, Y., and Taya, Y. (2000) p53AIP1, a potential mediator of p53-dependent apoptosis, and its regulation by Ser-46-phosphorylated p53, *Cell 102*, 849-862.

241. Stambolic, V., MacPherson, D., Sas, D., Lin, Y., Snow, B., Jang, Y., Benchimol, S., and Mak, T. W. (2001) Regulation of PTEN transcription by p53, *Mol. Cell 8*, 317-325.

242. Snøve, O, Jr, and Holen, T. (2004) Many commonly used siRNAs risk off-target activity, *Biochem. Biophys. Res. Commun. 319*, 256-263.

243. Chang, C., Norris, J. D., Grøn, H., Paige, L. A., Hamilton, P. T., Kenan, D. J., Fowlkes, D., and McDonnell, D. P. (1999) Dissection of the LXXLL nuclear receptor-coactivator interaction motif using combinatorial peptide libraries: discovery of peptide antagonists of estrogen receptors alpha and beta, *Mol. Cell. Biol. 19*, 8226-8239.

244. Zhang, X. K., Lehmann, J., Hoffmann, B., Dawson, M. I., Cameron, J., Graupner, G., Hermann, T., Tran, P., and Pfahl, M. (1992) Homodimer formation of retinoid X receptor induced by 9-cis retinoic acid, *Nature 358*, 587-591.

245. Allenby, G., Bocquel, M. T., Saunders, M., Kazmer, S., Speck, J., Rosenberger, M., Lovey, A., Kastner, P., Grippo, J. F., and Chambon, P. (1993) Retinoic acid receptors and retinoid X receptors: interactions with endogenous retinoic acids, *Proc Natl Acad Sci U S A 90*, 30-34.

246. Zhang, X. K., Salbert, G., Lee, M. O., and Pfahl, M. (1994) Mutations that alter ligand-induced switches and dimerization activities in the retinoid X receptor, *Mol. Cell. Biol. 14*, 4311-4323.

247. Chen, Z. P., Shemshedini, L., Durand, B., Noy, N., Chambon, P., and Gronemeyer, H. (1994) Pure and functionally homogeneous recombinant retinoid X receptor, *J Biol Chem 269*, 25770-25776.

248. Leid, M. (1994) Ligand-induced alteration of the protease sensitivity of retinoid X receptor alpha, *J Biol Chem 269*, 14175-14181.

249. Cheng, L., Norris, A. W., Tate, B. F., Rosenberger, M., Grippo, J. F., and Li, E. (1994) Characterization of the ligand binding domain of human retinoid X receptor alpha expressed in Escherichia coli, *J Biol Chem 269*, 18662-18667.

250. Chen, Z. P., Iyer, J., Bourguet, W., Held, P., Mioskowski, C., Lebeau, L., Noy, N., Chambon, P., and Gronemeyer, H. (1998) Ligand- and DNA-induced dissociation of RXR tetramers, *J Mol Biol 275*, 55-65.

251. Chen, Y., Wei, L., and Muller, J. D. (2003) Probing protein oligomerization in living cells with fluorescence fluctuation spectroscopy, *Proc Natl Acad Sci U S A 100*, 15492-15497.

252. Lala, D. S., Mukherjee, R., Schulman, I. G., Koch, S. S., Dardashti, L. J., Nadzan, A. M., Croston, G. E., Evans, R. M., and Heyman, R. A. (1996) Activation of specific RXR heterodimers by an antagonist of RXR homodimers, *Nature 383*, 450-453.

253. Allegretto, E. A., McClurg, M. R., Lazarchik, S. B., Clemm, D. L., Kerner, S. A., Elgort, M. G., Boehm, M. F., White, S. K., Pike, J. W., and Heyman, R. A. (1993) Transactivation properties of retinoic acid and retinoid X receptors in mammalian cells and yeast. Correlation with hormone binding and effects of metabolism, *J Biol Chem 268*, 26625-26633.

254. Vivat, V., Zechel, C., Wurtz, J. M., Bourguet, W., Kagechika, H., Umemiya, H., Shudo, K., Moras, D., Gronemeyer, H., and Chambon, P. (1997) A mutation mimicking ligand-induced conformational change yields a constitutive RXR that senses allosteric effects in heterodimers, *EMBO J. 16*, 5697-5709.

255. van Tilborg, P. J., Mulder, F. A., de Backer, M. M., Nair, M., van Heerde, E. C., Folkers, G., van der Saag, P. T., Karimi-Nejad, Y., Boelens, R., and Kaptein, R. (1999) Millisecond to microsecond time scale dynamics of the retinoid X and retinoic acid receptor DNA-binding domains and dimeric complex formation, *Biochemistry 38*, 1951-1956.

256. Bourguet, W., Ruff, M., Bonnier, D., Granger, F., Boeglin, M., Chambon, P., Moras, D., and Gronemeyer, H. (1995) Purification, functional characterization, and crystallization of the ligand binding domain of the retinoid X receptor, *Protein Expr. Purif. 6*, 604-608.

257. Kersten, S., Pan, L., and Noy, N. (1995) On the role of ligand in retinoid signaling: positive cooperativity in the interactions of 9-cis retinoic acid with tetramers of the retinoid X receptor, *Biochemistry 34*, 14263-14269.

258. Kersten, S., Pan, L., Chambon, P., Gronemeyer, H., and Noy, N. (1995) Role of ligand in retinoid signaling. 9-cis-retinoic acid modulates the oligomeric state of the retinoid X receptor, *Biochemistry 34*, 13717-13721.

259. Kersten, S., Dawson, M. I., Lewis, B. A., and Noy, N. (1996) Individual subunits of heterodimers comprised of retinoic acid and retinoid X receptors interact with their ligands independently, *Biochemistry 35*, 3816-3824.

260. Li, C., Schwabe, J. W., Banayo, E., and Evans, R. M. (1997) Coexpression of nuclear receptor partners increases their solubility and biological activities, *Proc Natl Acad Sci U S A 94*, 2278-2283.

261. Kersten, S., Gronemeyer, H., and Noy, N. (1997) The DNA binding pattern of the retinoid X receptor is regulated by ligand-dependent modulation of its oligomeric state, *J Biol Chem 272*, 12771-12777.

262. Kersten, S., Dong, D., Lee, W., Reczek, P. R., and Noy, N. (1998) Auto-silencing by the retinoid X receptor, *J Mol Biol 284*, 21-32.

263. Schimerlik, M. I., Peterson, V. J., Hobbs, P. D., Dawson, M. I., and Leid, M. (1999) Kinetic and thermodynamic analysis of 9-cis-retinoic acid binding to retinoid X receptor alpha, *Biochemistry 38*, 6732-6740.

264. Budhu, A. S., and Noy, N. (2000) On the role of the carboxyl-terminal helix of RXR in the interactions of the receptor with ligand, *Biochemistry 39*, 4090-4095.

265. Egea, P. F., Rochel, N., Birck, C., Vachette, P., Timmins, P. A., and Moras, D. (2001) Effects of ligand binding on the association properties and conformation in solution of retinoic acid receptors RXR and RAR, *J Mol Biol 307*, 557-576.

266. Egea, P. F., and Moras, D. (2001) Purification and crystallization of the human RXRalpha ligand-binding domain-9-cisRA complex, *Acta Crystallogr. D Biol. Crystallogr. 57*, 434-437.

267. Xu, Y., Fang, F., Chu, Y., Jones, D., and Jones, G. (2002) Activation of transcription through the ligand-binding pocket of the orphan nuclear receptor ultraspiracle, *Eur J Biochem 269*, 6026-6036.

268. Yan, X., Broderick, D., Leid, M. E., Schimerlik, M. I., and Deinzer, M. L. (2004) Dynamics and ligand-induced solvent accessibility changes in human retinoid x receptor homodimer determined by hydrogen deuterium exchange and mass spectrometry, *Biochemistry 43*, 909-917.

269. Harder, M. E., Deinzer, M. L., Leid, M. E., and Schimerlik, M. I. (2004) Global analysis of three-state protein unfolding data, *Protein Sci. 13*, 2207-2222.

270. Cavasotto, C. N., Liu, G., James, S. Y., Hobbs, P. D., Peterson, V. J., Bhattacharya, A. A., Kolluri, S. K., Zhang, X., Leid, M., Abagyan, R., Liddington, R. C., and Dawson, M. I. (2004) Determinants of retinoid X receptor transcriptional antagonism, *J Med Chem 47*, 4360-4372.

271. Yasmin, R., Yeung, K. T., Chung, R. H., Gaczynska, M. E., Osmulski, P. A., and Noy, N. (2004) DNA-looping by RXR Tetramers Permits Transcriptional Regulation "at a Distance", *J Mol Biol 343*, 327-338.

272. Lengqvist, J., Alvelius, G., Jornvall, H., Sjovall, J., Perlmann, T., and Griffiths, W. J. (2005) Electrospray mass spectrometry for the direct accurate mass measurement of ligands in complex with the retinoid X receptor alpha ligand binding domain, *J Am Soc Mass Spectrom 16*, 1631-1640.

273. Lengqvist, J., Mata de Urquiza, A., Perlmann, T., Sjovall, J., and Griffiths, W. J. (2005) Specificity of receptor-ligand interactions and their effect on dimerisation as observed by electrospray mass spectrometry: bile acids form stable adducts to the RXRalpha, *J Mass Spectrom 40*, 1448-1461.

274. Stebbins, J. L., Jung, D., Leone, M., Zhang, X. K., and Pellecchia, M. (2006) A structure-based approach to retinoid X receptor-alpha inhibition, *J Biol Chem 281*, 16643-16648.

275. Yan, X., Deinzer, M. L., Schimerlik, M. I., Broderick, D., Leid, M. E., and Dawson, M. I. (2006) Investigation of ligand interactions with human RXRalpha by hydrogen/deuterium exchange and mass spectrometry, *J Am Soc Mass Spectrom 17*, 1510-1517.

276. Folkertsma, S., van Noort, P. I., de Heer, A., Carati, P., Brandt, R., Visser, A., Vriend, G., and de Vlieg, J. (2007) The use of in vitro peptide binding profiles and in silico ligand-receptor interaction profiles to describe ligand-induced conformations of the retinoid X receptor alpha ligand-binding domain, *Mol. Endocrinol. 21*, 30-48.

277. Vivat-Hannah, V., Bourguet, W., Gottardis, M., and Gronemeyer, H. (2003) Separation of retinoid X receptor homo- and heterodimerization functions, *Mol. Cell. Biol. 23*, 7678-7688.

278. Mossakowska, D. E. (1998) Expression of nuclear hormone receptors in Escherichia coli, *Curr. Opin. Biotechnol. 9*, 502-505.

279. Halling, B. P., Yuhas, D. A., Eldridge, R. R., Gilbey, S. N., Deutsch, V. A., and Herron, J. D. (1999) Expression and purification of the hormone binding domain of the Drosophila ecdysone and ultraspiracle receptors, *Protein Expr. Purif. 17*, 373-386.

280. Rymarczyk, G., Grad, I., Rusek, A., Oswiecimska-Rusin, K., Niedziela-Majka, A., Kochman, M., and Ozyhar, A. (2003) Purification of Drosophila melanogaster ultraspiracle protein and analysis of its A/B region-dependent dimerization behavior in vitro, *Biol. Chem. 384*, 59-69.

281. Marks, M. S., Hallenbeck, P. L., Nagata, T., Segars, J. H., Appella, E., Nikodem, V. M., and Ozato, K. (1992) H-2RIIBP (RXR beta) heterodimerization provides a mechanism for combinatorial diversity in the regulation of retinoic acid and thyroid hormone responsive genes, *EMBO J. 11*, 1419-1435.

282. Medin, J. A., Minucci, S., Driggers, P. H., Lee, I. J., and Ozato, K. (1994) Quantitative increases in DNA binding affinity and positional effects determine 9-cis retinoic acid induced activation of the retinoid X receptor beta homodimer, *Mol. Cell. Endocrinol. 105*, 27-35.

283. Stafslien, D. K., Vedvik, K. L., De Rosier, T., and Ozers, M. S. (2007) Analysis of ligand-dependent recruitment of coactivator peptides to RXRbeta in a time-resolved fluorescence resonance energy transfer assay, *Mol. Cell. Endocrinol. 264*, 82-89.

284. Porath, J., Carlsson, J., Olsson, I., and Belfrage, G. (1975) Metal chelate affinity chromatography, a new approach to protein fractionation, *Nature 258*, 598-599.

285. Pace, C. N., Vajdos, F., Fee, L., Grimsley, G., and Gray, T. (1995) How to measure and predict the molar absorption coefficient of a protein, *Protein Sci. 4*, 2411-2423.

286. Nakajima, S., Yanagihara, I., and Ozono, K. (1997) A 65-kilodalton nuclear protein binds to the human vitamin D receptor: a bacterial-expressed histidine-tagged receptor study, *Biochem. Biophys. Res. Commun. 232*, 806-809.

287. Thomas, J. G., and Baneyx, F. (1997) Divergent effects of chaperone overexpression and ethanol supplementation on inclusion body formation in recombinant Escherichia coli, *Protein Expr. Purif. 11*, 289-296.

288. Watkins, R. E., Wisely, G. B., Moore, L. B., Collins, J. L., Lambert, M. H., Williams, S. P., Willson, T. M., Kliewer, S. A., and Redinbo, M. R. (2001) The human nuclear xenobiotic receptor PXR: structural determinants of directed promiscuity, *Science 292*, 2329-2333.

289. Gillespie, A. S., and Elliott, E. (2005) Comparative advantages of imidazole-sodium dodecyl sulfate-zinc reverse staining in polyacrylamide gels, *Anal. Biochem. 345*, 158-160.

290. Kenig, M., Peternel, S., Gaberc-Porekar, V., and Menart, V. (2006) Influence of the protein oligomericity on final yield after affinity tag removal in purification of recombinant proteins, *J Chromatogr A 1101*, 293-306.

291. Ventura, S., and Villaverde, A. (2006) Protein quality in bacterial inclusion bodies, *Trends Biotechnol. 24*, 179-185.

292. Low, L. Y., Hernandez, H., Robinson, C. V., O'Brien, R., Grossmann, J. G., Ladbury, J. E., and Luisi, B. (2002) Metal-dependent folding and stability of nuclear hormone receptor DNA-binding domains, *J Mol Biol 319*, 87-106.

293. Chen, Y., Wei, L., and Muller, J. D. (2005) Unraveling protein-protein

interactions in living cells with fluorescence fluctuation brightness analysis, *Biophys J 88*, 4366-4377.

294. Sanchez-Andres, A., Chen, Y., and Muller, J. D. (2005) Molecular brightness determined from a generalized form of Mandel's Q-parameter, *Biophys J 89*, 3531-3547.

295. Wright, E., Vincent, J., and Fernandez, E. J. (2007) Thermodynamic characterization of the interaction between CAR-RXR and SRC-1 peptide by isothermal titration calorimetry, *Biochemistry 46*, 862-870.

296. Redfern, C. P., and Wilson, K. E. (1993) Ligand binding properties of human cellular retinoic acid binding protein II expressed in E. coli as a glutathione-S-transferase fusion protein, *FEBS Lett. 321*, 163-168.

297. Gundersen, T. E., Lundanes, E., and Blomhoff, R. (1997) Quantitative high-performance liquid chromatographic determination of retinoids in human serum using on-line solid-phase extraction and column switching. Determination of 9-cis-retinoic acid, 13-cis-retinoic acid, all-trans-retinoic acid, 4-oxo-all-trans-retinoicacid and 4-oxo-13-cis-retinoic acid, *J Chromatogr B Biomed Sci Appl 691*, 43-58.

298. Scott, D. J., and Schuck, P. (2005) A brief introduction to the analytical ultracentrifugation of proteins for beginners, in *Analytical Ultracentrifugation: Techniques And Methods* (Scott, D. J., Harding, S. E., and Rowe, A. J., Eds.) pp 1-25, The Royal Society of Chemistry, Cambridge, UK.

299. Philo, J. S. (2000) Sedimentation equilibrium analysis of mixed associations using numerical constraints to impose mass or signal conservation, *Methods Enzymol. 321*, 100-120.

300. Laue, T. M., Shah, B. D., Ridgeway, T. M., and Pellieier, S. L. (1992) Computer-aided interpretation of sedimentation data for proteins, in (Harding, S. E. and Rowe, A. J., Eds.) pp 90-125, Royal Society of Chemistry, Cambridge, UK.

301. Santoro, M. M., and Bolen, D. W. (1988) Unfolding free energy changes determined by the linear extrapolation method. 1. Unfolding of phenylmethanesulfonyl alpha-chymotrypsin using different denaturants, *Biochemistry 27*, 8063-8068.

302. Ionescu, R. M., and Eftink, M. R. (1997) Global analysis of the acid-induced and urea-induced unfolding of staphylococcal nuclease and two of its variants, *Biochemistry 36*, 1129-1140.

303. Santoro, M. M., and Bolen, D. W. (1992) A test of the linear extrapolation of unfolding free energy changes over an extended denaturant concentration range, *Biochemistry 31*, 4901-4907.

304. Yao, M., and Bolen, D. W. (1995) How valid are denaturant-induced

unfolding free energy measurements? Level of conformance to common assumptions over an extended range of ribonuclease A stability, *Biochemistry 34*, 3771-3781.

305. Bolen, D. W., and Yang, M. (2000) Effects of guanidine hydrochloride on the proton inventory of proteins: implications on interpretations of protein stability, *Biochemistry 39*, 15208-15216.

306. Nelder, C. J., and Mead, R. (1965) A simplex method for function minization, *Comp. J. 7*, 308-313.

307. Senear, D. F., and Bolen, D. W. (1992) Simultaneous analysis for testing of models and parameter estimation, *Methods Enzymol. 210*, 463-481.

308. Beechem, J. M. (1992) Global analysis of biochemical and biophysical data, *Methods Enzymol. 210*, 37-54.

309. Straume, M. (1994) Sequential versus simultaneous analysis of data: differences in reliability of derived quantitative conclusions, *Methods Enzymol. 240*, 89-121.

310. Ramsay, G. D., and Eftink, M. R. (1994) Analysis of multidimensional spectroscopic data to monitor unfolding of proteins, *Methods Enzymol. 240*, 615-645.

311. Straume, M., and Johnson, M. L. (1992) Analysis of residuals: criteria for determining goodness-of-fit, *Methods Enzymol. 210*, 87-105.

312. Johnson, M. L. (1992) Why, when, and how biochemists should use least squares, *Anal. Biochem. 206*, 215-225.

313. Johnson, M. L., and Faunt, L. M. (1992) Parameter estimation by least-squares methods, *Methods Enzymol. 210*, 1-37.

314. Shoemaker, D. P., Garland, C. W., and Nibler, J. W. (1996) *Experiments in physical chemistry* 6th Ed., McGraw-Hill, Boston, Massachusetts.

315. Boehm, M. F., Zhang, L., Badea, B. A., White, S. K., Mais, D. E., Berger, E., Suto, C. M., Goldman, M. E., and Heyman, R. A. (1994) Synthesis and structure-activity relationships of novel retinoid X receptor-selective retinoids, *J Med Chem 37*, 2930-2941.

316. Lewis, M. S., and Youle, R. J. (1986) Ricin subunit association. Thermodynamics and the role of the disulfide bond in toxicity, *J Biol Chem 261*, 11571-11577.

317. Vistica, J., Dam, J., Balbo, A., Yikilmaz, E., Mariuzza, R. A., Rouault, T. A., and Schuck, P. (2004) Sedimentation equilibrium analysis of protein interactions with global implicit mass conservation constraints and systematic noise decomposition, *Anal. Biochem. 326*, 234-256.

318. Sanchez-Ruiz, J. M., Lopez-Lacomba, J. L., Cortijo, M., and Mateo, P. L.

(1988) Differential scanning calorimetry of the irreversible thermal denaturation of thermolysin, *Biochemistry 27*, 1648-1652.

319. Sanchez-Ruiz, J. M. (1992) Theoretical Analyasis of Lumry-Eyring Models in Differential Scanning Calorimetry, *Biophys J 61*, 935.

320. Eftink, M. R. (1994) The use of fluorescence methods to monitor unfolding transitions in proteins, *Biophys J 66*, 482-501.

321. Eftink, M. R. (1998) The use of fluorescence methods to monitor unfolding transitions in proteins, *Biochemistry (Moscow) 63*, 276-284.

322. Royer, C. A. (2006) Probing protein folding and conformational transitions with fluorescence, *Chem Soc Rev 106*, 1769-1784.

323. Silverman, J. A., and Harbury, P. B. (2002) Rapid mapping of protein structure, interactions, and ligand binding by misincorporation proton-alkyl exchange, *J Biol Chem 277*, 30968-30975.

324. Munson, P. J., and Rodbard, D. (1983) Number of receptor sites from Scatchard and Klotz graphs: a constructive critique, *Science 220*, 979-981.

325. Scatchard, G. (1948) The attraction of proteins for small molecules and ions., *Ann. N. Y. Acad. Sci. 51*, 660-672.

326. Cogan, U., Kopelman, M., Mokady, S., and Shinitzky, M. (1976) Binding affinities of retinol and related compounds to retinol binding proteins, *Eur J Biochem 65*, 71-78.

327. Johnson, M. L. (2000) Outliers and robust parameter estimation, *Methods Enzymol. 321*, 417-424.

328. Disdier, B., Bun, H., Catalin, J., and Durand, A. (1996) Simultaneous determination of all-trans-, 13-cis-, 9-cis-retinoic acid and their 4-oxo-metabolites in plasma by high-performance liquid chromatography, *J Chromatogr B Biomed Appl 683*, 143-154.

329. Adams, P. D., Chen, Y., Ma, K., Zagorski, M. G., Sonnichsen, F. D., McLaughlin, M. L., and Barkley, M. D. (2002) Intramolecular quenching of tryptophan fluorescence by the peptide bond in cyclic hexapeptides, *J Am Chem Soc 124*, 9278-9286.

330. Callis, P. R., and Liu, T. Q. (2004) Quantitative prediction of fluorescence quantum yields for tryptophan in proteins, *J. Phys. Chem. B 108*, 4248-4259.

331. Greenfield, N., Vijayanathan, V., Thomas, T. J., Gallo, M. A., and Thomas, T. (2001) Increase in the stability and helical content of estrogen receptor alpha in the presence of the estrogen response element: analysis by circular dichroism spectroscopy, *Biochemistry 40*, 6646-

6652.

332. Kallen, J., Schlaeppi, J., Bitsch, F., Delhon, I., and Fournier, B. (2004) Crystal structure of the human RORalpha Ligand binding domain in complex with cholesterol sulfate at 2.2 A, *J Biol Chem 279*, 14033-14038.

333. Yu, C., Chen, L., Luo, H., Chen, J., Cheng, F., Gui, C., Zhang, R., Shen, J., Chen, K., Jiang, H., and Shen, X. (2004) Binding analyses between Human PPARgamma-LBD and ligands, *Eur J Biochem 271*, 386-397.

334. Nair, S. K., Thomas, T. J., Greenfield, N. J., Chen, A., He, H., and Thomas, T. (2005) Conformational dynamics of estrogen receptors alpha and beta as revealed by intrinsic tryptophan fluorescence and circular dichroism, *J Mol Endocrinol 35*, 211-223.

335. Ortlund, E. A., Bridgham, J. T., Redinbo, M. R., and Thornton, J. W. (2007) Crystal structure of an ancient protein: evolution by conformational epistasis, *Science 317*, 1544-1548.

336. Shkolny, D. L., Beitel, L. K., Ginsberg, J., Pekeles, G., Arbour, L., Pinsky, L., and Trifiro, M. A. (1999) Discordant measures of androgen-binding kinetics in two mutant androgen receptors causing mild or partial androgen insensitivity, respectively, *J Clin Endocrinol Metab 84*, 805-810.

337. Imasaki, K., Hasegawa, T., Okabe, T., Sakai, Y., Haji, M., Takayanagi, R., and Nawata, H. (1994) Single amino acid substitution (840Arg-->His) in the hormone-binding domain of the androgen receptor leads to incomplete androgen insensitivity syndrome associated with a thermolabile androgen receptor, *Eur J Endocrinol 130*, 569-574.

338. Askew, E. B., Gampe, R. T. J., Stanley, T. B., Faggart, J. L., and Wilson, E. M. (2007) Modulation of androgen receptor activation function 2 by testosterone and dihydrotestosterone, *J Biol Chem 282*, 25801-25816.

339. Fox, R., Roy, A., Govindarajan, S., Minshull, J., Gustafsson, C., Jones, J. T., and Emig, R. (2003) Optimizing the search algorithm for protein engineering by directed evolution, *Protein Eng. 16*, 589-597.

340. Fox, R. (2005) Directed molecular evolution by machine learning and the influence of nonlinear interactions, *J Theor Biol 234*, 187-199.

341. Fox, R. J., Davis, S. C., Mundorff, E. C., Newman, L. M., Gavrilovic, V., Ma, S. K., Chung, L. M., Ching, C., Tam, S., Muley, S., Grate, J., Gruber, J., Whitman, J. C., Sheldon, R. A., and Huisman, G. W. (2007) Improving catalytic function by ProSAR-driven enzyme evolution, *Nat. Biotechnol. 25*, 338-344.

342. Wells, J. A. (1990) Additivity of mutational effects in proteins, *Biochemistry 29*, 8509-8517.

343. Gregoret, L. M., and Sauer, R. T. (1993) Additivity of mutant effects assessed by binomial mutagenesis, *Proc Natl Acad Sci U S A 90*, 4246-4250.

344. Thomas, H. E., Stunnenberg, H. G., and Stewart, A. F. (1993) Heterodimerization of the Drosophila ecdysone receptor with retinoid X receptor and ultraspiracle, *Nature 362*, 471-475.

345. La Vista-Picard, N., Hobbs, P. D., Pfahl, M., Dawson, M. I., and Pfahl, M. (1996) The receptor-DNA complex determines the retinoid response: a mechanism for the diversification of the ligand signal, *Mol. Cell. Biol. 16*, 4137-4146.

346. Quack, M., and Carlberg, C. (2000) Ligand-triggered stabilization of vitamin D receptor/retinoid X receptor heterodimer conformations on DR4-type response elements, *J Mol Biol 296*, 743-756.

347. Makinen, J., Reinisalo, M., Niemi, K., Viitala, P., Jyrkkarinne, J., Chung, H., Pelkonen, O., and Honkakoski, P. (2003) Dual action of oestrogens on the mouse constitutive androstane receptor, *Biochem J. 376*, 465-472.

348. Stanley, T. B., Leesnitzer, L. M., Montana, V. G., Galardi, C. M., Lambert, M. H., Holt, J. A., Xu, H. E., Moore, L. B., Blanchard, S. G., and Stimmel, J. B. (2003) Subtype specific effects of peroxisome proliferator-activated receptor ligands on corepressor affinity, *Biochemistry 42*, 9278-9287.

349. Poujol, N., Margeat, E., Baud, S., and Royer, C. A. (2003) RAR antagonists diminish the level of DNA binding by the RAR/RXR heterodimer, *Biochemistry 42*, 4918-4925.

350. Lew, J. L., Zhao, A., Yu, J., Huang, L., De Pedro, N., Pelaez, F., Wright, S. D., and Cui, J. (2004) The farnesoid X receptor controls gene expression in a ligand- and promoter-selective fashion, *J Biol Chem 279*, 8856-8861.

351. Geistlinger, T. R., McReynolds, A. C., and Guy, R. K. (2004) Ligand-Selective Inhibition of the Interaction of Steroid Receptor Coactivators and Estrogen Receptor Isoforms, *Chem. Biol. 11*, 273-281.

352. Nettles, K. W., Sun, J., Radek, J. T., Sheng, S., Rodriguez, A. L., Katzenellenbogen, J. A., Katzenellenbogen, B. S., and Greene, G. L. (2004) Allosteric control of ligand selectivity between estrogen receptors alpha and beta: implications for other nuclear receptors, *Mol. Cell 13*, 317-327.

353. Schaufele, F., Carbonell, X., Guerbadot, M., Borngraeber, S., Chapman, M. S., Ma, A. A., Miner, J. N., and Diamond, M. I. (2005) The structural basis of androgen receptor activation: intramolecular and

intermolecular amino-carboxy interactions, *Proc Natl Acad Sci U S A 102*, 9802-9807.

354. Schulman, I. G., Li, C., Schwabe, J. W., and Evans, R. M. (1997) The phantom ligand effect: allosteric control of transcription by the retinoid X receptor, *Genes Dev. 11*, 299-308.

355. Pissios, P., Tzameli, I., Kushner, P., and Moore, D. D. (2000) Dynamic stabilization of nuclear receptor ligand binding domains by hormone or corepressor binding, *Mol. Cell 6*, 245-253.

356. Borngraeber, S., Budny, M. J., Chiellini, G., Cunha-Lima, S. T., Togashi, M., Webb, P., Baxter, J. D., Scanlan, T. S., and Fletterick, R. J. (2003) Ligand selectivity by seeking hydrophobicity in thyroid hormone receptor, *Proc Natl Acad Sci USA 100*, 15358-15363.

357. Madauss, K. P., Deng, S. J., Austin, R. J., Lambert, M. H., McLay, I., Pritchard, J., Short, S. A., Stewart, E. L., Uings, I. J., and Williams, S. P. (2004) Progesterone receptor ligand binding pocket flexibility: crystal structures of the norethindrone and mometasone furoate complexes, *J Med Chem 47*, 3381-3387.

358. Nettles, K. W., Bruning, J. B., Gil, G., O'neill, E. E., Nowak, J., Guo, Y., Kim, Y., Desombre, E. R., Dilis, R., Hanson, R. N., Joachimiak, A., and Greene, G. L. (2007) Structural plasticity in the oestrogen receptor ligand-binding domain, *EMBO Rep. 8*, 610.

359. Bohl, C. E., Wu, Z., Miller, D. D., Bell, C. E., and Dalton, J. T. (2007) Crystal structure of the T877A human androgen receptor ligand-binding domain complexed to cyproterone acetate provides insight for ligand-induced conformational changes and structure-based drug design, *J Biol Chem 282*, 13648-13655.

360. Johnson, B. A., Wilson, E. M., Li, Y., Moller, D. E., Smith, R. G., and Zhou, G. (2000) Ligand-induced stabilization of PPARgamma monitored by NMR spectroscopy: implications for nuclear receptor activation, *J Mol Biol 298*, 187-194.

361. Shulman, A. I., Larson, C., Mangelsdorf, D. J., and Ranganathan, R. (2004) Structural determinants of allosteric ligand activation in RXR heterodimers, *Cell 116*, 417-429.

362. Folkertsma, S., van Noort, P., Van Durme, J., Joosten, H. J., Bettler, E., Fleuren, W., Oliveira, L., Horn, F., de Vlieg, J., and Vriend, G. (2004) A family-based approach reveals the function of residues in the nuclear receptor ligand-binding domain, *J Mol Biol 341*, 321-335.

363. de Felipe, K. S., Carter, B. T., Althoff, E. A., and Cornish, V. W. (2004) Correlation between ligand-receptor affinity and the transcription readout in a yeast three-hybrid system, *Biochemistry 43*, 10353-10363.

364. Bergman, T., Henrich, V. C., Schlattner, U., and Lezzi, M. (2004) Ligand control of interaction in vivo between ecdysteroid receptor and ultraspiracle ligand-binding domain, *Biochem J. 378*, 779-784.

365. Dimitriadis, E. K., and Lewis, M. S. (2000) Optimal data analysis using transmitted light intensities in analytical ultracentrifuge, *Methods Enzymol. 321*, 121-136.

366. Lewis, M. S., and Reily, M. M. (2005) Application of weighted robust regression to equilibrium ultracentrifugation, in *Analytical Ultracentrifugation: Techniques And Methods* (Scott, D. J., Harding, S. E., and Rowe, A. J., Eds.) pp 152-161, The Royal Society of Chemistry, Cambridge, UK.

367. Lewis, M. S., Reily, M. M., and Holladay, L. A. (2005) Fitting of thermodynamic data from equilibrium ultracentrifugation by robust least-squares with reduced parameter cross-correlation coefficients, in *Analytical Ultracentrifugation: Techniques And Methods* (Scott, D. J., Harding, S. E., and Rowe, A. J., Eds.) pp 372-388, The Royal Society of Chemistry, Cambridge, UK.

368. Royer, C. A., and Beechem, J. M. (1992) Numerical analysis of binding data: advantages, practical aspects, and implications, *Methods Enzymol. 210*, 481-505.

369. Phillips, G. R., and Eyring, E. M. (1988) Error Estimation Using the Sequential Simplex-Method in Nonlinear Least-Squares Data-Analysis, *Anal. Chem. 60*, 738-741.

370. Brumby, S. (1989) Exchange of Comments on the Simplex Algorithm Culminating in Quadratic Convergence and Error Estimation, *Anal. Chem. 61*, 1783-1786.

371. Johnson, L. A. (1992) Analysis of variance of parameter estimates: F tests and t tests, *Anal. Biochem. 206*, 195-201.

372. Straume, M., and Johnson, M. L. (1992) Monte Carlo method for determining complete confidence probability distributions of estimated model parameters, *Methods Enzymol. 210*, 117-129.

373. Johnson, M. L. (2000) Parameter correlations while curve fitting, *Methods Enzymol. 321*, 424-446.

374. Di Cera, E. (1992) Use of weighting functions in data fitting, *Methods Enzymol. 210*, 68-87.

375. Demeler, B. (2005) Ultrascan: a comprehensive data analysis software package for analytical ultracentrifugation experiments, in *Analytical Ultracentrifugation: Techniques And Methods* (Scott, D. J., Harding, S. E., and Rowe, A. J., Eds.) pp 210-229, The Royal Society of Chemistry, Cambridge, UK.